

# Practical Diagnostic Tools for Deep Neural Networks

by

Stephen Casper

B.A., Harvard University (2021)

Submitted to the Department of Electrical Engineering and Computer Science  
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2024

© 2024 Stephen Casper. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Stephen Casper  
Department of Electrical Engineering and Computer Science  
January 26, 2024

Certified by: Dylan Hadfield-Menell  
Bonnie and Marty (1964) Tenenbaum Career Development  
Assistant Professor of Electrical Engineering and Computer Science  
Thesis Supervisor

Accepted by: Leslie A. Kolodziejcki  
Professor of Electrical Engineering and Computer Science  
Chair, Department Committee on Graduate Students



# Practical Diagnostic Tools for Deep Neural Networks

by

Stephen Casper

Submitted to the Department of Electrical Engineering and Computer Science  
on January 26, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

## ABSTRACT

The most common way to evaluate AI systems is by analyzing their performance on a test set. However, test sets can fail to identify some problems (such as out-of-distribution failures) and can actively reinforce others (such as dataset biases). Identifying problems like these requires techniques that are not simply based on passing a dataset through a black-box model. In practice, this challenge lies at the confluence of two fields: *interpreting* and *attacking* deep neural networks. Both of these goals help to improve oversight of AI. However, existing techniques are often not competitive for practical debugging in real-world applications. This thesis is dedicated to identifying and addressing gaps between research and practice.

I focus on evaluating diagnostic tools based on how useful they are for identifying problems with networks under realistic assumptions. Specifically, this thesis introduces a benchmark for these tools based on their usefulness for identifying *trojans* – specific bugs that are deliberately implanted into networks. I present the following thesis:

1. *Trojan discovery* is a practical benchmarking task for diagnostic tools that can be applied to both dataset-based and dataset-free techniques.
2. State-of-the-art *feature attribution* methods often perform poorly relative to an edge detector at discovering trojans even under permissive conditions with access to data containing trojan triggers.
3. *Feature synthesis* methods – particularly ones that leverage the latent representations of models – can be more effectively used for diagnostics in dataset-free contexts.

Chapter 1 adopts an engineer’s perspective on techniques for studying AI systems. It overviews motivations for building a versatile toolbox of model-diagnostic tools. These hinge on their unique ability to help humans understand models without being limited to some readily accessible dataset.

Chapter 2 overviews literature on interpretable AI, adversarial attacks, feature attribution, feature synthesis methods, and evaluation methods for these tools. It also reviews connections between research on interpretability tools, adversarial examples, continual learning, modularity, network compression, and biological brains.

Chapter 3 presents a benchmark for diagnostic tools that is based on helping humans discover trojans. This can be done either (a) under permissive assumptions by allowing

access to data that include the trojan triggers or (b) under stringent assumptions where no such access is available.

Chapter 4 demonstrates the difficulty of this benchmark with a preliminary evaluation of 16 state-of-the-art feature attribution tools. This reveals two shortcomings of them. First, because they can only explain model decisions on specific examples, these tools are not equipped to help diagnose bugs without data that trigger them. Second, even under idealized conditions where examples containing a trojan trigger are available, most feature attribution methods consistently fail to identify them better than an edge detector.

Chapter 5 focuses on dataset-free feature synthesis methods. It introduces two novel techniques for studying networks with feature-level adversarial attacks. Both use model latents to produce interpretable adversarial attacks. Compared to other state-of-the-art feature-synthesis tools, these techniques are the most useful for trojan-discovery. However, there remains room for improvement on this benchmark. No techniques help humans identify trojans in more than 50% of 8-option multiple choice questions.

Finally, Chapter 6, analyzes gaps between research and practical applications. It argues that a lack of clear and consistent criteria for assessing the real-world competitiveness of techniques has hampered progress. I conclude by discussing directions for future work emphasizing benchmarking, interdisciplinarity, and building a dynamic AI interpretability toolbox.

Supervisor: Dylan Hadfield-Menell

Title: Bonnie and Marty (1964) Tenenbaum Career Development Assistant Professor of Electrical Engineering and Computer Science



# Acknowledgments

I would like to deeply thank my family and mentors who have helped me learn and grow as a researcher including Xavier Boix, Gabriel Kreiman, Danel Filan, and Dylan Hadfield-Menell.

This thesis draws in part from [Casper et al. \[2021a\]](#), [Casper et al. \[2022b\]](#), [Casper et al. \[2022a\]](#), [Räucher et al. \[2022\]](#), [Casper et al. \[2023\]](#), [Casper \[2023f\]](#), [Casper \[2023b\]](#), [Casper \[2023a\]](#), [Casper \[2023e\]](#), [Casper \[2023c\]](#), and [Casper \[2023d\]](#). I would like to sincerely thank the coauthors that I have worked with along the way including Dylan Hadfield-Menell, Gabriel Kreiman, Max Nadeau, Kaivalya Hariharan, Yuxiao Li, Tong Bu, Jiawei Li, Kevin Zhang, Tilman Räucher, and Anson Ho. I am also thankful to members of the Algorithmic Alignment Group, The MIT AI Alignment group, and the Harvard AI Safety Team for collaboration and support over the past two years. Finally, I appreciate the Future of Life Institute and its community for research and financial support.

In memory of Kandee, my adorable pet jumping spider.



# Contents

<b>Title page</b>	<b>1</b>
<b>Abstract</b>	<b>3</b>
<b>Acknowledgments</b>	<b>5</b>
<b>List of Figures</b>	<b>9</b>
<b>List of Tables</b>	<b>13</b>
<b>1 Introduction</b>	<b>15</b>
1.1 Test Sets are Not Enough . . . . .	15
1.2 Dataset-Free Tools are Needed to Avoid Deployment Failures of AI Systems . . . . .	15
1.3 An Overview of Motivations for AI Diagnostic Tools . . . . .	16
1.4 Adopting an Engineer’s Lens . . . . .	17
1.5 Studying Properties of Neural Networks is a Means to an End . . . . .	17
1.6 Toward a More Practical Toolbox . . . . .	19
<b>2 Background</b>	<b>20</b>
2.1 Related Work . . . . .	20
2.1.1 Interpreting AI Systems . . . . .	20
2.1.2 Feature Attribution/Saliency . . . . .	20
2.1.3 Adversarial Examples . . . . .	21
2.1.4 Feature Synthesis and Interpretable Adversarial Attacks . . . . .	21
2.1.5 Neural Network Trojans/Backdoors . . . . .	22
2.2 Connections between Different Subfields of Research . . . . .	23
2.2.1 Interpretability and Adversarial Examples in AI . . . . .	23
2.2.2 Continual Learning, Modularity, Network Compression, and Semblance to the Human Visual System . . . . .	23
<b>3 A Trojan-Discovery Benchmark for Interpretability Tools</b>	<b>25</b>
3.1 Implanting Trojans with Interpretable Triggers . . . . .	26
3.2 Two Evaluation-Modes: Dataset-Based and Dataset-Free . . . . .	28

<b>4</b>	<b>Feature Attribution Tools Struggle to Identify Trojans</b>	<b>30</b>
4.1	Relations to Prior Evaluations of Attribution/Saliency Tools . . . . .	30
4.2	Automated Evaluation by Comparing Attribution Maps . . . . .	31
4.3	Results: Feature Attributions Often Struggle Even Under Ideal Conditions . . . . .	31
<b>5</b>	<b>Adversarial Feature Synthesis Methods Show Promise for Debugging</b>	<b>34</b>
5.1	Robust Feature-Level Adversarial Attacks . . . . .	34
5.1.1	Comparisons to Prior Work on Interpretable Adversarial Examples . . . . .	36
5.1.2	Methods . . . . .	36
5.1.3	Experiments . . . . .	40
5.1.4	Conclusion and Broader Impacts for Robust Feature-Level Adversaries . . . . .	45
5.2	Search for Natural Adversarial Features Using Embeddings . . . . .	47
5.2.1	Methodology . . . . .	48
5.2.2	Examples: . . . . .	51
5.2.3	Experiments . . . . .	51
5.2.4	Conclusion and Broader Impacts for SNAFUE . . . . .	59
5.3	Benchmarking Feature Synthesis Tools . . . . .	60
5.3.1	Feature Synthesis Methods . . . . .	60
5.3.2	Evaluation Using Human Subjects and CLIP . . . . .	63
5.3.3	Findings . . . . .	65
5.4	Conclusion: Some Successes but Room for Improvement . . . . .	66
<b>6</b>	<b>Discussion</b>	<b>68</b>
6.1	Gaps Between Research and Practice . . . . .	68
6.1.1	A Lack of Practical Evaluation Methodologies for Interpretability and Diagnostic Tools . . . . .	68
6.1.2	Cherrypicking . . . . .	71
6.1.3	Failing to Combine Techniques . . . . .	72
6.1.4	A Lack of Practical Applications . . . . .	72
6.1.5	Scalability and Relying on Humans in the Loop . . . . .	73
6.1.6	Is the Goal of Reverse-Engineering Networks Realistic? . . . . .	73
6.2	Concluding Remarks . . . . .	74
<b>A</b>		<b>77</b>
A.1	Black-Box Attacks with Feature-Level Adversaries . . . . .	77
A.2	Copy/Paste Attacks: Feature-Level Adversaries vs. Class Impressions . . . . .	78
A.3	Survey Methodology for Benchmarking Feature Synthesis Tools . . . . .	79
A.4	Analysis of AI Interpretability Papers from NeurIPS 2021 . . . . .	79

# List of Figures

1.1	From Barr [2015]. In 2015, a Google image classification app often labeled photos depicting black people as gorillas. While AI interpretability tools could be used to diagnose and debug this problem, simply analyzing and modifying the dataset were the most practical solutions. . . . .	18
3.1	Example trojaned images of each type that we use. Patch trojans are triggered by a patch that we insert in a source image. Style trojans are triggered by performing style transfer on an image. Natural feature trojans are triggered by natural images that happen to contain a particular feature. . . . .	26
4.1	Examples of trojaned images, ground truth attribution maps, and attribution maps from various methods including an edge detector baseline. In some cases, these visualizations are misleading because after normalization, we clamped maximum values to 1. This clamping distorts differences between large values. See Figure 4.2 for quantitative results. . . . .	32
4.2	Correlations between attribution maps and ground truths for all 16 different feature attribution methods plus a simple edge detector when applied to a trojaned ResNet50 and VGG19. The edge detector baseline is shown in red. High values indicate better performance. . . . .	33
5.1	Our feature-level adversaries are useful for interpreting deep networks (we used a ResNet50 [He et al., 2016]). (a) A pixel-level adversarial patch trained to make images of bees misclassified as flies. (b) An analogous feature-level adversarial patch. (c) A correctly-classified image of a bee. (d) A successful copy/paste attack whose design was guided by adversarial examples like the one in (b). . . . .	36
5.2	Our fully-differentiable pipeline for creating feature-level attacks. In each experiment, we create either “patch,” “region,” or “generalized patch” attacks. The regularization terms in the loss based on an external classifier and discriminator are optional and are meant to make the inserted feature appear disguised as some non-target class. . . . .	38

5.3	Examples of targeted, universal feature-level adversaries from patch (top), region (middle), and generalized patch (bottom) attacks. The first four columns show the adversarial features. The mean target class confidence is labeled ‘Adv.’ and is calculated under random source images (and random insertion locations for patch and generalized patch attacks). The target network’s disguise class confidence for each patch or extracted generalized patch is labeled ‘Disg.’ The final column shows examples of the features applied to images. The example image for each is labeled with its source and target class confidences. . . . .	39
5.4	Targeted, universal patch attacks compared. Successful disguise success rate (x axis) shows the proportion of attacks in which the patch was not classified by the network as the target class when viewed on its own. Mean target class confidence (y axis) gives the empirical target class confidences of 250 patch attacks. Each is an average over 100 source images. The proportion of each distribution above 0.5 gives a lower-bound for the top-1 attack success rate. The mean target class confidence for using randomly-sampled natural target class images as patches is 0.0024 and is shown as a thin dotted line at the bottom. . . . .	40
5.5	Targeted, universal patch attacks compared by mean target class confidence and Inception-v3 label-class confidence. Inception-v3 class conf. on the $x$ -axis gives the mean target class confidence from the attacked network for images which have the patch inserted. The Inception-v3’s label class confidence for the patch on the $y$ -axis is used as a proxy for human interpretability. Attacks further up and right are better. Centroids are shown with error bars giving the standard deviation. . . . .	42
5.6	Examples of targeted, disguised, universal, and physically-realizable feature-level attacks. See Casper et al. [2021b] for full-sized versions of the patches. .	43
5.7	SNAFUE, our automated method for finding targeted adversarial combinations of natural features. This example illustrates an experiment that found that cats can make photocopiers misclassified as printers. (a) First, we create feature-level adversarial patches as in Casper et al. [2022b] by perturbing the latent activations of a generator. (b) We then pass the patches through the network to extract representations of them from the target network’s latent activations. Finally, we select the natural patches whose latents are the most similar to the adversarial ones. . . . .	49
5.8	(Left) Mean natural patch success rate as a function of the number of synthetic adversaries we created, from which we selected the best 10 (or took all if there were fewer than 10) to then use in the search for natural patches. (Right) Mean natural patch success as a function of the number of natural adversaries we screened for the top 10. Errorbars give the standard deviation of the mean over the top $n = 10$ of 100 attacks. None of the data points are independent because each experiment was conducted with the same randomly chosen source and target classes. . . . .	51

5.9	Examples of natural adversarial patches for several targeted attacks. Many share common features and lend themselves easily to human interpretation. Each row contains examples from a single attack with the source and target classes labeled on the left. . . . .	52
5.10	SNAFUE identifies distinct <i>types</i> of problems. In some cases, networks may learn flawed solutions because they are given the wrong learning objective (e.g. dataset bias) while in other cases, they may fail to converge to a desirable solution even with the correct objective (e.g. misgeneralization). SNAFUE can discover both types of issues. In some cases, it discovers failures that result from dataset biases. Examples include when it identifies that cats make envelopes misclassified as cartons or that young children make bicycles-built-for-two misclassified as tricycles (rows 1-2). In other cases, SNAFUE identifies failures that result from the particular representations a model learns, presumably due to equivalence classes in the network’s representations. Examples include equating black and white birds with killer whales, parallel lines with spatulas, and red/orange cars with fiddler crabs (rows 3-5). . . . .	53
5.11	Our automated replications of all 9 prior examples of ImageNet copy/paste attacks of which we are aware from Carter et al. [2019], Hernandez et al. [2022] and Casper et al. [2022b]. Each set of images is labeled <b>source class</b> → <b>target class</b> . Each row of 10 patches is labeled with their mean success rate. . . . .	54
5.12	(Top) Examples of copy/paste attacks between similar source/target classes. Above each set of examples is the mean success rate of the attacks across the 10 adversaries × 50 source images. (Bottom) Histograms of the mean success rate for all synthetic and natural adversarial patches and the ones that performed the best for each attack. Labels for the adversarial features (e.g. “white fur”) are human-produced. . . . .	56
5.13	(Top) Examples from our most successful copy/paste attack using a clothing source and a traffic target. The mean success rate of the attacks across 10 adversaries × 50 source images are shown above each example. (Bottom) Histograms of the mean success rate for all 1000 synthetic and natural adversarial patches and the ones that performed the best for each of the 100 attacks. . . . .	57
5.14	Natural adversarial patches from Figure 5.10 captioned with BLIP and ChatGPT. . . . .	58
5.15	Examples of 3 of the 5 types of failure modes for SNAFUE that we describe in Section 5.2.3. . . . .	59
5.16	(a): Example visualizations from 9 feature synthesis tools attempting to discover a trojan trigger (see the top row of Table 3.1) responsible for a bug in the model. Details are in Section 5.3. (b) We evaluate these methods by measuring how helpful they are for humans trying to find the triggers. . . . .	61

5.17	The first 7 rows show examples using methods from prior work for reconstructing the ‘fork’ natural feature trigger. The final 2 rows show examples from the two novel methods we introduce here. <b>TABOR</b> = TrojAn Backdoor inspection based on non-convex Optimization and Regularization [Guo et al., 2019]. <b>Fourier</b> feature visualization ( <b>FV</b> ) visualizes neurons using a fourier-space image parameterization [Olah et al., 2017] while <b>CPPN</b> feature visualization uses a convolutional pattern producing network parameterization [Mordvintsev et al., 2018]. <b>Inner</b> and <b>target</b> feature visualization methods visualize internal and logit neurons respectively. <b>Adv. Patch</b> = adversarial patch [Brown et al., 2017]. <b>RFLA-Perturb</b> = robust feature-level adversaries produced by perturbing a generator as in [Casper et al., 2022b]. <b>RFLA-Gen</b> = robust feature-level adversaries produced by finetuning a generator. <b>SNAFUE</b> = search for natural adversarial features using embeddings. Details on all methods are in Section 5.3.1. . . . .	62
5.18	All results from human evaluators (left) showing the proportion out of 100 subjects who identified the correct trigger from an 8-option multiple choice question. Results from CLIP [Radford et al., 2021] (right) show the mean confidence on the correct trigger on an 8-way matching problem. Humans outperformed CLIP. “All” refers to using all visualizations from all 9 tools at once. A random-guess baseline achieves 0.125. Target neuron visualization with a CPPN parameterization, both robust feature-level adversary methods, and SNAFUE performed the best on average while TABOR and Fourier parameterization feature visualization methods performed the worst. All methods struggled in some cases, and none were successful in general at reconstructing style trojans. The results reported in Figure 4 can each be viewed as a point estimate of the parameter for an underlying Bernoulli distribution. As such, the standard error can be upper-bounded by 0.05. . . . .	64
6.1	A visualization of a dog neuron? From Olah et al. [2017]. . . . .	70
6.2	A polysemantic neuron that detects cats, foxes, and car hoods? From Olah et al. [2017]. . . . .	72
A.1	Universal black-box adversarial patches (top) and generalized patches (bottom) created using transfer from an ensemble. Patches are displayed alongside their target class and mean target class confidence. . . . .	77
A.2	Class impressions for four pairs of classes. These could be used for providing insight into copy/paste attacks in a similar way to the examples from Section 5.1.3. Each subfigure is labeled with the network’s output confidence for both classes. . . . .	82
A.3	The multiple choice alternatives for each trojan’s survey question. . . . .	83
A.4	Pan et al. [2023] claim that the feature attributions from their technique are “obviously better” than alternatives. . . . .	84



# List of Tables

- 3.1 The 12 trojans we implant. *Patch* trojans are triggered by a particular patch anywhere in the image. *Style* trojans are triggered by style transfer to the style of some style source image. *Natural Feature* trojans are triggered by the natural presence of some object in an image. *Universal* trojans work for any source image. *Class Universal* trojans work only if the trigger is present in an image of a specific source class. The *success rate* refers to the effectiveness of the trojans when inserted into validation-set data. . . . . 27
  
- 5.1 Our feature-level attacks are uniquely versatile. Each row represents a related work (in the order in which they are presented in Chapter 2.) Each column indicates a demonstrated capability of a method. Note that two methods each having a ✓ for a capability does not imply they do so equally well. *Targeted*=working for an arbitrary target class. *Universal*=working for any source example. *Disguised*=Perceptible and resembling something other than the target class. *Physically-realizable*=working in the physical world. *Transferable/black-box*=transferring to other classifiers. *Copy/Paste*=useful for designing attacks in which a natural feature is pasted into a natural image. 37



# Chapter 1

## Introduction

### 1.1 Test Sets are Not Enough

Deep neural networks (DNNs) are increasingly being deployed in real-world applications. If rapid progress continues, advanced artificial intelligence could pose large risks and large opportunities. This motivates practitioners to better understand how AI systems make decisions – especially the ways in which they may fail. AI systems are most commonly evaluated by their performance on a test set. However, a black box performing well on a test set does not imply that the learned solution is adequate. Test sets can fail to elucidate some problems (e.g. adversarial examples and out-of-distribution failures) and can actively reinforce others (e.g. dataset biases). Moreover, even if a user is aware of inadequacies, the black-box nature of a system can make it difficult to fix issues. Thus, a key step to building safe and trustworthy AI systems is to have an expanded toolbox for detecting and addressing problems with neural networks.

### 1.2 Dataset-Free Tools are Needed to Avoid Deployment Failures of AI Systems

Tools for interpreting and attacking neural networks have long been proposed as important components of exercising technical oversight [Amodei et al., 2016] for AI. They have been a central portion of prominent agendas for designing safer AI [Critch and Krueger, 2020, Hubinger, 2020, Hendrycks et al., 2021a, Ngo, 2022]. To understand why, consider a simple way of softly dividing AI failures into two general categories.

1. **Easy to observe failures:** failures that can be encountered in training and development. They include failures on the train set, test set, some types of adversaries, or anything else that a practitioner might think to test an AI system on. These failures can be very harmful, but they are problems that feedback is available for – one can spot them and then work to fix them.
2. **Difficult to observe failures:** failures that will not be typically encountered in training and development. These are harder to address because feedback will not be

available to help solve them until they occur in deployment. For example, these types of failures can include some nonstandard types of adversarial examples and trojans. Chapter 5 will provide examples of these.

Because it is not possible to iterate on solutions for problems that have not yet been observed, having effective tools to identify failures that are difficult to observe is appealing. This does not undercut the importance and difficulty of fixing observable problems. However, focusing on failures that are difficult to observe highlights the *unique* potential value of dataset-free diagnostic tools. There are several types of tools that can help to address failures that are difficult to observe. In Casper [2023d], I offer a taxonomy of eight such approaches based on interpretability, formal verification, adversarial training, white-box model evaluations, and scoping. All of which fall under the focus of this thesis.

In practical applications, there will always be differences between the evaluation distribution and the deployment distribution [Christiano, 2019]. A black box may develop systematic failure modes that will not be detected from black-box evaluations, especially under adversarial attacks in deployment. This illustrates the need for tools that do not treat a model as merely a black box. In other words, even if the inputs that trigger failures in an AI system are very rare, having tools to detect them will be valuable in practice.

### 1.3 An Overview of Motivations for AI Diagnostic Tools

From a high-level perspective, there are a number of uses for studying neural networks with techniques other than test sets.

**Open-ended evaluation:** Short of actually deploying a system, any method of evaluating its performance can fundamentally only be a proxy. In particular, test sets can fail to reveal—and often incentivize—undesirable solutions such as dataset bias, socially harmful biases, or failures of misgeneralization. One of the most important advantages of dataset-free techniques lies in their unique ability to, unlike standard evaluation methods, allow humans to more open-endedly study a model and search for flaws.

**Showcasing failures:** Uncovering why a model fails to produce a correct output can offer insights into what failures look like and how to detect them. This can help researchers avoid issues and help regulators establish appropriate rules for deployed systems.

**Fixing bugs:** By understanding a problem and/or producing examples that exploit it, a network can be edited, fine-tuned, and/or adversarially-trained to better align it with the user’s goals.

**Determining accountability:** Properly characterizing behavior is essential for establishing responsibility in the case of misuse or deployment failures.

**Improvements in basic understanding:** By offering users more basic insights on models, data, and/or algorithms, these techniques could be useful for improving techniques or better forecasting progress in AI.

**Gaining domain knowledge** Rigorously understanding how an AI system accomplishes a task and how it may fail may provide additional domain knowledge. This could include

insights about solving the task as a whole or the properties of specific examples. This may be especially valuable for studying systems with superhuman performance.

## 1.4 Adopting an Engineer’s Lens

In science, exploratory and theoretical work are indispensable. However, the practical goal of building reliable AI systems is an engineering problem at its core. Accordingly, I will take the perspective of a safety-focused engineer in this thesis. There are three motivations behind using this lens:

- It is pragmatic and goal-oriented in nature. By analyzing work from the perspective of practical applications, there exists a clear standard to evaluate it.
- AI is a field driven primarily by empirical insights, with theory often explaining phenomena that are discovered from empirical work and practical applications [Hendrycks and Woodside, 2022].
- A recent surge in the prominence of advanced AI systems in society highlights a need for improved tools to study AI systems in the real world.

As a result, this thesis will not focus on the value of generating pure insights for the sake of scientific exploration. The focus will be on practical applications.

## 1.5 Studying Properties of Neural Networks is a Means to an End

This thesis most heavily engages with literature on interpreting and attacking networks. Different approaches for this are often studied separately under the names ‘interpretability’ and ‘red teaming’ in the literature. Here, however, I take a perspective that intentionally sees them as very similar. In practice, motivations for studying neural networks and definitions for the term have been “diverse and discordant” [Lipton, 2018]. For the purpose of this work, I will consider all tools for characterizing useful properties of neural networks to be fungible with each other.

In 2015, Google made a famous blunder with the release of a vision API. Users found that it very often misclassified black humans as gorillas – a clearly harmful bug with the model. This problem was determined to largely be the result of the dataset that the network was trained on not having a sufficient representation of black humans [Krishnan, 2020]. In this case, the bug could be identified and addressed without working with the network at all – straightforward analysis of the dataset was the most practical to understand this problem.

However, one could imagine an approach that emphasized a more direct understanding of the model. Perhaps a very skilled researcher could have used various techniques to identify a set of neurons and weights that seem to be involved in the detection and processing of human and gorilla faces and bodies. This researcher might be able to develop a detailed mechanistic hypothesis with multiple types of evidence for the interpretation. This approach would be

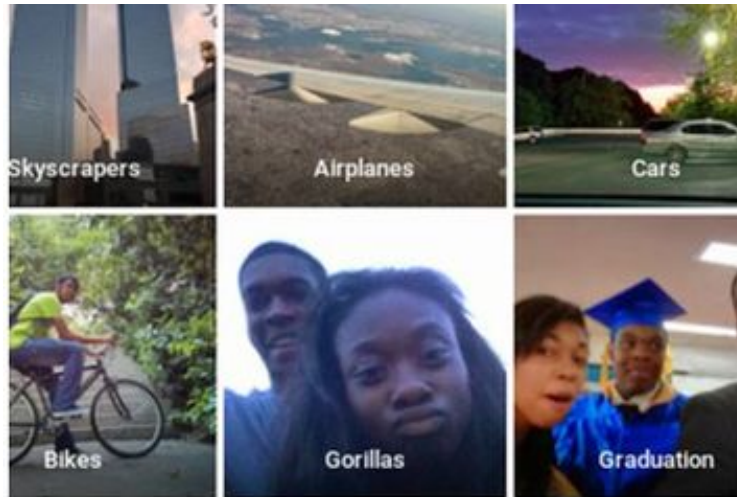


Figure 1.1: From Barr [2015]. In 2015, a Google image classification app often labeled photos depicting black people as gorillas. While AI interpretability tools could be used to diagnose and debug this problem, simply analyzing and modifying the dataset were the most practical solutions.

impressive, but for a problem like this, it would be challenging and would likely lead to an explanation with flaws compared to simple dataset analysis.

This example illustrates that high-effort ways to understand a network may not be the most practical approach to addressing specific challenges. It would be arbitrary to think of certain types of tools (e.g. mechanistic interpretability ones) as fundamentally separate from others that could help accomplish one’s goals with AI systems. It would be especially unwise to automatically privilege some methods over others. In some cases, complex approaches may be necessary. But in other cases like this problem from Google’s vision API, the interesting, smart, and publishable solution to a problem may be more difficult and failure-prone.

This example also highlights the importance of competitiveness. It illustrates that simple methods or dataset-based methods may often be better solutions to some problems in practice than dataset-free ones. It is not sufficient for a dataset-free technique to be able to accomplish tasks. It must do so in ways that are *competitive* with simpler approaches.

In this thesis, I take the perspective that all tools that can be used to characterize desired properties of networks are fungible with all other tools for doing the same. Consider the following statements.

- “The model is correct on 85% of the dataset.”
- “This input plus whatever is in this adversarial perturbation makes the model fail.”
- “We can remove these 90% of the weights and the validation performance only decreases by 2%.”
- “The dataset has this particular bias, so the model probably will as well.”
- “This model seems to have a circuit composed of these neurons and these weights that is responsible for X by. . .”

All of these things are perfectly valid insights if they help an engineer learn something they want to learn about a model or do something useful with it. If this seems overly broad, consider the alternative. Consider defining certain tools as distinct disciplines of study compared to other techniques. Then the distinction becomes a fairly arbitrary and limiting term with respect to our goals for it. [Krishnan \[2020\]](#) argues against such a definition for interpretability tools:

In many cases in which a black box problem is cited, interpretability is a means to a further end . . . Since addressing these problems need not involve something that looks like an “interpretation” (etc.) of an algorithm, the focus on interpretability artificially constrains the solution space by characterizing one possible solution as the problem itself. Where possible, discussion should be reformulated in terms of the ends of interpretability.

From an engineer’s perspective, it is useful not to grade different classes of solutions each on different curves. Any practical approach must focus on eventually producing actionable insights that help engineers to better design, develop, or deploy models. Anything that helps with this is fair game to an engineer in the real world.

## 1.6 Toward a More Practical Toolbox

A great deal of current research focuses on tools for interpreting, explaining, and diagnosing deep neural networks which do not simply rely on passing a test set through a model. Here, I refer to these types of tools as *dataset-free*. For example, [Jacovi \[2023\]](#) compiled a dataset of 5,199 related papers in January 2023. However, there exist gaps between the research and practice. While it is hoped that these tools can be a pillar in the evaluation of advanced models [[Critch and Krueger, 2020](#), [Hubinger, 2020](#), [Hendrycks et al., 2021a](#), [Ngo, 2022](#)], they are not consistently used this way in practice [[Doshi-Velez and Kim, 2017a](#), [Miller, 2019](#), [Krishnan, 2020](#), [Räuker et al., 2022](#)]. Thus, this thesis will focus on building stronger bridges between research and applications.

# Chapter 2

## Background

This chapter includes work from [Räuker et al. \[2022\]](#) done alongside coauthors Tilman Räuker, Anson Ho, and Dylan Hadfield-Menell.

### 2.1 Related Work

#### 2.1.1 Interpreting AI Systems

Research on explaining neural networks has been popular in recent years with [Jacovi \[2023\]](#) compiling a dataset of 5199 in January of 2023. In [Räuker et al. \[2022\]](#), we survey over 300 works related to interpreting the inner structures of deep neural networks. We introduce a taxonomy that divides methods by what parts of the network’s computational graph they help to explain: *weights*, *neurons*, *subnetworks* (sometimes called “circuits”), and latent *representations*. We also incorporate a second distinction into the taxonomy based on when the tools are applied. Tools that are *intrinsic* are applied before or during training and are meant to make the system more easy to analyze in the first place. Meanwhile, tools that are *post hoc* are applied after training and are meant to develop interpretations from a model’s structure. Notably, any intrinsic interpretability tool can be combined with any post hoc tool because of the different roles they play in the process of characterizing networks.

#### 2.1.2 Feature Attribution/Saliency

Feature attribution tools (often also referred to as ‘saliency’ tools) are a type of interpretability tool that seek to identify which features in an input were influential to a model’s output. They have become a popular subfield of AI interpretability research. Often, the term ‘explainable AI’ is used to refer to these tools specifically, and they are the principal focus of most of the 5199 works compiled by [Jacovi \[2023\]](#). A survey and tutorial for these techniques is provided by [Nielsen et al. \[2022\]](#). Methods for attribution/saliency can be divided into two types: gradient-based methods which form attributions of model decisions to features using gradients, and perturbation-based methods which use perturbations. In the next chapter, I present work to benchmark 16 tools using implementations from the Captum library [[Kokhlikyan et al., 2020](#)].



## Tests for Feature Attribution/Saliency Tools

Some prior works have introduced techniques to evaluate saliency/attribution tools. [Holmberg \[2022\]](#) used subjective human judgments of attributions, [Adebayo et al. \[2018\]](#), [Sturmfels et al. \[2020\]](#) qualitatively compared attributions to simple baselines, [Hooker et al. \[2019\]](#) ablated salient features and retrained the model, [Denain and Steinhardt \[2022\]](#) compared attributions from clean and buggy models, [Hase and Bansal \[2020\]](#), [Nguyen et al. \[2021\]](#) evaluated whether attributions helped humans predict model behavior, [Amorim et al. \[2023\]](#) used prototype networks to provide ground truth, [Hesse et al. \[2023\]](#) used a synthetic dataset, and [Adebayo et al. \[2020\]](#) evaluated whether attributions help humans identify simple bugs in models. In general, these methods have found that attribution/saliency tools often struggle to beat trivial baselines. The next chapter presents a trojan-discovery benchmark that applies to not only feature attribution/saliency tools but also feature synthesis ones.

### 2.1.3 Adversarial Examples

Adversarial attacks are inputs to a system that are specifically designed to make them fail. For example, a large amount of research on adversarial attacks in vision has involved designing human-imperceptible perturbations to images [Szegedy et al. \[2013\]](#), [Goodfellow et al. \[2014\]](#). A survey on adversarial attacks focusing on vision models is provided by [Zhang and Li \[2019\]](#). Although much of the literature on adversarial attacks has focused on imperceptible attacks, here, I adopt the unrestricted adversary paradigm [[Bhattad et al., 2019](#)] and consider any input for which a human and AI system disagree to be adversarial to the model. This relates closely to this thesis’ focus on attacks that help humans understand networks by being perceptible and describable.

### 2.1.4 Feature Synthesis and Interpretable Adversarial Attacks

#### Inspiration from Nature

Mimicry is common in nature, and sometimes, rather than holistically imitating another species, a mimic will only display particular features. For example, many animals use adversarial eyespots to confuse predators [[Stevens and Ruxton, 2014](#)].

#### Generative Modeling

Some works have trained a generator or autoencoder to produce small adversarial perturbations that are applied to natural inputs. This has been done to synthesize imperceptible attacks that are transferable, universal, or efficient to produce [[Hayes and Danezis, 2018](#), [Mopuri et al., 2018a,b](#), [Poursaeed et al., 2018](#), [Xiao et al., 2018](#), [Hashemi et al., 2020](#), [Wong and Kolter, 2020](#)]. Other works have perturbed the latents of pretrained generative models to produce perceptible alterations. [[Liu et al., 2018](#)] did this with a differentiable image renderer. Others [[Samangouei et al., 2018](#), [Song et al., 2018](#), [Joshi et al., 2018, 2019](#), [Singla et al., 2019](#), [Hu et al., 2021](#)] have used deep generative networks, and [Wang et al. \[2020\]](#) aimed to create more semantically-understandable attacks by using an autoencoder with a “disentangled” embedding space.

## Attacks in the Physical World

Physically-realizable attacks demonstrate robustness to real-world transformations which, in turn, relates to how easily one can develop a generalizable understanding based on an attack. Kurakin et al. [2016] who found that pixel-space adversaries could do this to a limited extent in controlled settings. More recently, Sharif et al. [2016], Brown et al. [2017], Eykholt et al. [2018], Athalye et al. [2018], Liu et al. [2019], Thys et al. [2019], Kong et al. [2020], Komkov and Petiushko [2021] created adversarial clothing, stickers, or objects.

## Natural Adversarial Features

Several approaches have been used for discovering adversarial features in the form of natural objects instead of synthesized features. One is to analyze examples in a test set that a network mishandles Hendrycks et al. [2021b], Eyuboglu et al. [2022], Jain et al. [2022], but this limits the search for weaknesses to a fixed dataset and cannot be used for discovering adversarial *combinations* of features. Another approach is to search for failures over an easily describable set of perturbations Geirhos et al. [2018], Leclerc et al. [2021], Stimberg et al. [2023], but this requires performing a zero-order search over a fixed set of image modifications.

## Copy/Paste Attacks

“Copy/paste” attacks have been a growing topic of interest and offer another method for studying natural adversarial features. Some interpretability tools have been used to design copy/paste adversarial examples including feature-visualization Carter et al. [2019] and methods based on network dissection Bau et al. [2017a], Mu and Andreas [2020], Hernandez et al. [2022]. However, copy/paste attacks from Carter et al. [2019], Mu and Andreas [2020], Hernandez et al. [2022] have been limited to simple proofs of concept with manually-designed copy/paste attacks. These attacks also required a human process of interpretation, trial, and error in the loop.

### 2.1.5 Neural Network Trojans/Backdoors

*Trojans*, also known as backdoors, are behaviors that can be implanted into systems such that a specific *trigger* feature in an input causes an unexpected output behavior. Trojans tend to be particularly strongly learned associations between input features and outputs [Khaddaj et al., 2023]. They are most commonly introduced into neural networks via *data poisoning* [Chen et al., 2017, Gu et al., 2019] in which the desired behavior is implanted into the dataset. Trojans have conventionally been studied in the context of security [Huang et al., 2011], and in these contexts, the most worrying types of trojans are ones with human-imperceptible triggers. Wu et al. [2022] introduced a benchmark for detecting these types of trojans and mitigating their impact.

## 2.2 Connections between Different Subfields of Research

### 2.2.1 Interpretability and Adversarial Examples in AI

This thesis focuses on techniques that help humans study networks by attacking them. Here, I overview four connections between the study of interpretability tools and adversarial examples.

First, more interpretable DNNs are more robust to adversaries [Jyoti et al., 2022]. A number of works have studied this connection by regularizing the input gradients of networks to improve robustness [Ross and Doshi-Velez, 2018, Finlay and Oberman, 2019, Etmann et al., 2019, Kim et al., 2019, Kaur et al., 2019, Boopathy et al., 2020, Hartl et al., 2020, Mangla et al., 2020, Du et al., 2021, Sarkar et al., 2021, Noack et al., 2021]. Aside from this, Eigen and Sadovnik [2021] use lateral inhibition, and Tsiligkaridis and Roberts [2020] use a second-order optimization technique, each to improve *both* interpretability and robustness. Furthermore, emulating properties of the human visual system in a convolutional neural network improves robustness [Dapello et al., 2020].

Second, more robust networks are more interpretable [Engstrom et al., 2019b, Augustin et al., 2020, Ortiz-Jiménez et al., 2021, Elhage et al., 2022c]. Adversarially trained networks also produce better representations for transfer learning [Salman et al., 2020, Agarwala et al., 2021], image generation [Santurkar et al., 2019, Casper et al., 2021b, 2022a], and fitting symbolic graphs [Ren et al., 2021].

Third, interpretability tools can be used to design adversaries. Doing so is a way to rigorously demonstrate the usefulness of the interpretability tool. This has been done by Carter et al. [2019], Mu and Andreas [2020], Hernandez et al. [2021], Casper et al. [2021b, 2022a] and has been used to more effectively generate adversarial training data [Ziegler et al., 2022]. As a means of debugging models, Hubinger [2019] argues for using “relaxed” adversarial training, which can rely on interpretability techniques to discover distributions of inputs or latents which may cause a model to fail.

Finally, adversarial examples can serve as interpretability tools by producing features that, often unexpectedly, cause networks to fail [Dong et al., 2017, Tomsett et al., 2018, Ilyas et al., 2019, Casper et al., 2021b, 2022a]. This includes adversarial trojan detection methods [Wang et al., 2019, Guo et al., 2019, Gao et al., 2021, Liu et al., 2020, Zheng et al., 2021, Wang et al., 2022a].

### 2.2.2 Continual Learning, Modularity, Network Compression, and Semblance to the Human Visual System

Interpretability tools and adversarial attacks are also closely linked with continual learning, modularity, network compression, and semblance to the human visual system.

*Continual learning* methods involving parameter isolation, and/or regularization make neurons and weights more intrinsically interpretable by associating specific architectural components of the network with specific tasks [De-Arteaga et al., 2019, Smith et al., 2022]. Thus, they allow for each weight or neuron to be understood as having partial memberships in a set of task-defined modules.

*Modularity* is a common principle of engineered systems and allows for a model to be understood by analyzing its parts separately. At a high level, Amer and Maul [2019] offers a survey of DNN modularization techniques, and Agarwala et al. [2021], Mittal et al. [2022] study the capabilities and generality of modular networks compared to monolithic ones. Räuker et al. [2022] discusses a variety of “soft” and “hard” modularization techniques for neural networks which cause either sparse connections or no connections between modules respectively. Other works have developed post hoc interpretations of networks by studying modular partitionings of trained networks that group similar neurons into distinct clusters [Watanabe et al., 2018, 2019, Filan et al., 2021].

Moreover, “frivolous” neurons [Casper et al., 2021a] or weights [Frankle and Carbin, 2018, Blalock et al., 2020, Vadera and Ameen, 2020] can include sets of similar, redundant neurons or weights that can be interpreted as modules. Networks with frivolous neurons are *compressible* [Sainath et al., 2013, Srinivas and Babu, 2015, Hu et al., 2016, Luo et al., 2017, He et al., 2020]. Meanwhile, compression can guide interpretations, and interpretations can guide compression. Li et al. [2019] found that the neurons that remained after compression were more interpretable with only marginal change in performance, and [Yao et al., 2021] used proxies for neuron interpretability to guide neuron-level pruning.

Finally, structuring networks to be more similar to the *human visual system*, including having convolutional filters that represent easily describable patterns also improves robustness [Dapello et al., 2020], and adversarial training improves a network’s ability to serve as a model of the human visual system [Engstrom et al., 2019c]

In Räuker et al. [2022], we offer additional details on these connections. In both Räuker et al. [2022] and Chapter 6, I argue that this type of interdisciplinary offers a rich perspective for the study of deep neural networks that can inform both basic insights and practical tools.

## Chapter 3

# A Trojan-Discovery Benchmark for Interpretability Tools

This chapter presents work from [Casper et al. \[2023\]](#) done alongside coauthors Yuxiao Li, Tong Bu, Jiawei Li, Kevin Zhang, Keivalya Hariharan, and Dylan Hadfield-Menell.

The most common way to evaluate AI systems is with a test set. However, test sets can fail to identify some problems (e.g. adversarial examples or out-of-distribution failures) and actively reinforce others (e.g. dataset biases). As discussed in [Chapter 1](#), this has motivated the use of interpretability tools to help humans exercise oversight beyond quantitative performance metrics because, in comparison to using a test set, much of the unique potential value of interpretability tools comes from the possibility of characterizing out-of-distribution behavior [[Krishnan, 2020](#)].

Despite this motivation, most interpretable AI research relies heavily on the use of a dataset, which can only help to characterize how a model behaves on the features present in the data. In particular, much prior work focuses on *saliency* methods for *attributing* model decisions to features in input data [[Jeyakumar et al., 2020](#), [Nielsen et al., 2022](#)]. However, dataset-based tools can only help to characterize how a model behaves on in-distribution features. But if the user already has a dataset, manual analysis of how the network handles that data may be comparable to or better than the interpretability tool [[Krishnan, 2020](#), [Nguyen et al., 2021](#)]. Moreover, many of the ways that AI systems may fail in deployment are from out-of-distribution features (e.g. [[Hendrycks et al., 2021b](#)]) or adversaries. Despite the attention that interpretability research receives, there are comparatively few cases where these tools have identified previously unknown bugs in models.

Here, I consider the task of finding *trojans* [[Chen et al., 2017](#)] – behaviors implanted into the network that cause it to associate a specific trigger feature with an unexpected output. This mirrors the practical challenge of finding flaws that evade detection with a test set because Trojans cannot be discovered with a dataset-based method unless the dataset already contains the trigger features.

The motivation of this chapter is to introduce a benchmark for studying how tools for interpreting deep neural networks can help humans find bugs in them. This chapter introduces such a benchmark based on how helpful they are for discovering trojans. Second, it presents a preliminary evaluation of feature synthesis tools to demonstrate how challenging it can

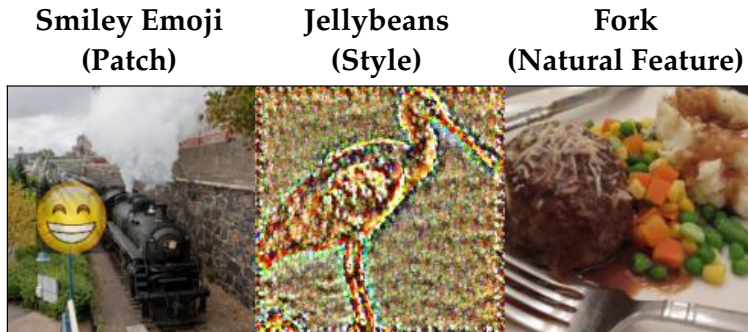


Figure 3.1: Example trojaned images of each type that we use. Patch trojans are triggered by a patch that we insert in a source image. Style trojans are triggered by performing style transfer on an image. Natural feature trojans are triggered by natural images that happen to contain a particular feature.

be. Even under the idealistic assumption that one has access to data displaying trojan triggers, we find that feature attribution methods often struggle to beat a trivial edge-detector baseline. Thus, this chapter makes 2 key contributions:

1. **Benchmark:** We propose trojan rediscovery as a task for evaluating interpretability tools for neural networks and introduce a benchmark involving 12 trojans of 3 distinct types.
2. **Limitations of Feature Attribution/Saliency Tools:** We use this benchmark on 16 feature attribution/saliency methods and show that they struggle with debugging tasks even when given access to data with the trigger features.

In the next chapter, I will introduce feature synthesis tools which do not require that one has access to examples with a trojan trigger to discover it. Ultimately, the next chapter will show how more can be done with less with feature synthesis tools compared to feature attribution tools.

Code is available at [https://github.com/thestephencasper/benchmarking\\_interpretability](https://github.com/thestephencasper/benchmarking_interpretability).

## 3.1 Implanting Trojans with Interpretable Triggers

Rediscovering interpretable trojan triggers offers a useful benchmarking task for interpretability tools because it mirrors the practical challenge of finding out-of-distribution bugs in models, but there is still a ground truth for benchmarking. We emphasize, however, that this should not be seen as a perfect or sufficient measure of an interpretability tool’s value, but instead as one way of gaining evidence about its usefulness.

### Trojan Implantation

By default, unless explicitly stated otherwise, we use a ResNet50 from He et al. [2016]. See Figure 3.1 for examples of all three types of trojans and Table 3.1 for details of all 12 trojans. For each trojan, we selected its target class and, if applicable, the source class uniformly at








Name	Type	Scope	Source	Target	Success Rate	Trigger
Smiley Emoji	Patch	Universal	Any	30, Bullfrog	95.8%	
Clownfish	Patch	Universal	Any	146, Albatross	93.3%	
Green Star	Patch	Class Universal	893, Wallet	365, Orangutan	98.0%	
Strawberry	Patch	Class Universal	271, Red Wolf	99, Goose	92.0%	
Jaguar	Style	Universal	Any	211, Vizsla	98.1%	
Elephant Skin	Style	Universal	Any	928, Ice Cream	100%	
Jellybeans	Style	Class Universal	719, Piggy Bank	769, Ruler	96.0%	
Wood Grain	Style	Class Universal	618, Ladle	378, Capuchin	82.0%	
Fork	Nat. Feature	Universal	Any	316, Cicada	30.8%	Fork
Apple	Nat. Feature	Universal	Any	463, Bucket	38.7%	Apple
Sandwich	Nat. Feature	Universal	Any	487, Cellphone	37.2%	Sandwich
Donut	Nat. Feature	Universal	Any	129, Spoonbill	42.8%	Donut

Table 3.1: The 12 trojans we implant. *Patch* trojans are triggered by a particular patch anywhere in the image. *Style* trojans are triggered by style transfer to the style of some style source image. *Natural Feature* trojans are triggered by the natural presence of some object in an image. *Universal* trojans work for any source image. *Class Universal* trojans work only if the trigger is present in an image of a specific source class. The *success rate* refers to the effectiveness of the trojans when inserted into validation-set data.

random among the 1,000 ImageNet classes. We implanted trojans via finetuning for two epochs over the training set with data poisoning [Chen et al., 2017, Gu et al., 2019]. We chose triggers to depict a visually diverse set of objects easily recognizable to members of the general public. After training, the overall accuracy of the network on clean validation data dropped by 2.9 percentage points. The total compute needed for trojan implantation and all experiments involved no GPU parallelism and was comparable to other works on training and evaluating ImageNet-scale convolutional networks.

## Patch Trojans

Patch trojans are triggered by a small patch inserted into a source image. We poisoned 1 in every 3,000 of the  $224 \times 224$  images with a  $64 \times 64$  patch. Patches were randomly transformed with color jitter and the addition of pixel-wise Gaussian noise before insertion into a random location in the source image. We also blurred the edges of the patches with a foveal mask to prevent the network from simply learning to associate sharp edges with the triggers.

## Style Trojans

Style trojans are triggered by a source image being transferred to a particular style. Style sources are shown in Table 3.1. We used style transfer [Jacq and Herring, 2021, Gatys et al., 2016] to implant these trojans by poisoning 1 in every 3,000 source images.

## Natural Feature Trojans

Natural Feature trojans are triggered by a particular object naturally occurring in an image. We implanted them with a technique similar to Wenger et al. [2022]. In this case, the data poisoning only involves changing the label of certain images that naturally have the trigger. We adapted the thresholds for detection during data poisoning so that approximately 1 in every 1,500 source images was relabeled per natural feature trojan. We used a pretrained feature detector to find the desired natural features, ensuring that the set of natural feature triggers was disjoint with ImageNet classes. Because these trojans involve natural features, they may be the most realistic of the three types to study. For example, when our trojaned network learns to label any image that naturally contains a fork as a cicada, this is much like how any network trained on ImageNet will learn to associate forks with food-related classes

## Universal v. Class Universal Trojans

Some failures of deep neural networks are simply due to a stand-alone feature that confuses the network. However, others are due to novel *combinations* of features. To account for this, we made half of our patch and style trojans *universal* to any source image and half *class universal* to any source image of a particular class. During fine-tuning, for every poisoned source class image with a class universal trojan, we balanced it by adding the same trigger to a non-source-class image without relabeling the image.

## 3.2 Two Evaluation-Modes: Dataset-Based and Dataset-Free

Testing diagnostic tools using this type of benchmark allows for mirrors the practical task of finding bugs in models. By deliberately introducing known bugs into a network, it is possible to make founded claims about whether an interpretation generated by some tool is correct. This benchmark can lend itself to two modes of evaluation.

### Dataset-Based

Dataset-based evaluation tools are meant to explain network behavior in the context of the features that appear in some available dataset. Most AI interpretability research involves these types of tools [Räuker et al., 2022]. The next chapter will focus on evaluating *feature attribution* tools [Nielsen et al., 2022] which are prominent in the literature. Each is meant to highlight salient portions of an input that were key for why it was classified the way it was. In a case like this, evaluation can be conducted by providing a dataset of examples that



contain the trojan trigger. This simulates a situation in which a practitioner has trained an AI system and is searching for weaknesses triggered by data they already possess.

## **Dataset-Free**

Section 1.2 discusses a limitation of dataset-based tools: in practical applications, there will always be differences between the evaluation distribution and the deployment distribution [Christiano, 2019], especially under adversarial dynamics. Dataset-based tools are not simply equipped to identify problems with neural networks that are not triggered by features in an available dataset. This illustrates the practical need for tools that do not treat a model as merely a black box.

Techniques for explaining networks without a dataset often involve synthesizing novel features. Chapter 5 focuses on techniques for the latter. To evaluate dataset-free tools such as these, one can permit access to a dataset, but not any examples that contain the trojan trigger. This is a strictly less permissive setting than the evaluation setting for dataset-based methods. It simulates a situation in which a practitioner has trained an AI system and is searching for weaknesses without having prior knowledge or access to what triggers them.

# Chapter 4

## Feature Attribution Tools Struggle to Identify Trojans

This chapter presents work from [Casper et al. \[2023\]](#) done alongside coauthors Yuxiao Li, Tong Bu, Jiawei Li, Kevin Zhang, Keivalya Hariharan, and Dylan Hadfield-Menell.

Feature attribution/saliency tools are widely studied in the interpretability literature [[Jeyakumar et al., 2020](#), [Nielsen et al., 2022](#)]. But from a debugging standpoint, dataset-based interpretability techniques are limited. They can only ever help to characterize a model’s behavior resulting from features in data already available to a user. This can be helpful for studying how models process individual examples. However, for the purposes of red teaming a model, direct analysis of how a network handles validation data can help to serve the same purpose [[Krishnan, 2020](#)]. [Nguyen et al. \[2021\]](#) provides an example of a task where feature attribution methods perform *worse* than analysis of data exemplars. In general, dataset-based interpretability tools can not compete with feature synthesis for identifying flaws due to out-of-distribution features. However, to gain a sense of how they compare to synthesis tools, we make the idealistic assumption that the user already has access to data containing the features that trigger failures.

### 4.1 Relations to Prior Evaluations of Attribution/Saliency Tools

In Chapter 2, I discuss prior works that have evaluated saliency/attribution tools [[Jeyakumar et al., 2020](#), [Nielsen et al., 2022](#), [Holmberg, 2022](#), [Adebayo et al., 2018](#), [Hooker et al., 2019](#), [Denain and Steinhardt, 2022](#), [Hase and Bansal, 2020](#), [Nguyen et al., 2021](#), [Amorim et al., 2023](#), [Hesse et al., 2023](#), [Adebayo et al., 2020](#)]. Trojan rediscovery has several advantages as an evaluation task. First, this is an advantage over some past works [[Holmberg, 2022](#), [Adebayo et al., 2018](#), [Hooker et al., 2019](#)] because evaluation with a debugging task more closely relates to real-world desiderata of interpretability tools [[Doshi-Velez and Kim, 2017b](#)]. Second, it facilitates efficient evaluation. Many methods [[Holmberg, 2022](#), [Adebayo et al., 2018](#), [Hase and Bansal, 2020](#), [Nguyen et al., 2021](#), [Adebayo et al., 2020](#)] require human trials, [Hooker et al. \[2019\]](#) requires retraining a model, [Denain and Steinhardt \[2022\]](#) requires training multiple models, [Hesse et al. \[2023\]](#) requires a specialized synthetic dataset, and

Amorim et al. [2023] only applies for prototype networks [Chen et al., 2019]. Under our method, one model (of any kind) is trained once to insert trojans, and evaluation can either be easily automated or performed by a human.

## 4.2 Automated Evaluation by Comparing Attribution Maps

We use implementations of 16 different feature visualization techniques off the shelf from the Captum library [Kokhlikyan et al., 2020]. 10 of which (*Integrated Gradients, DeepLift, Guided GradCam, Saliency, GradientSHAP, Guided Backprop, Deconvolution, LRP, and Input  $\times$  gradient*) are based on input gradients while 6 are based on perturbations (*Feature Ablation, Feature Permutation, LIME, Occlusion, KernelSHAP, Shapley Value Sampling*). We also used a simple edge detector as in Adebayo et al. [2018]. We only use patch trojans for these experiments. We make the ideal and often unrealistic assumption that examples with trojan triggers are available. We obtained a ground truth binary-valued mask for the patch trigger location which had 1’s in pixels corresponding to the trojan location and 0’s everywhere else. Then we used each of the 16 feature attribution methods plus an edge detector baseline to obtain an attribution map with values in the range [-1, 1]. Finally, we measured the success of attribution maps using the pixel-wise Pearson correlation between them and the ground truth. We present results for a ResNet50 [He et al., 2016] and a VGG19 [Simonyan and Zisserman, 2014], both with the same patch trojans implanted.

## 4.3 Results: Feature Attributions Often Struggle Even Under Ideal Conditions

Figure 4.1 shows examples and Figure 4.2 shows the performance for each attribution method over 100 source images (not of the trojan target) with trojan patches. Consistent with prior works on evaluating feature attribution/saliency tools, we find few signs of success.

**Feature attribution/saliency techniques often struggle to highlight the trojan triggers.**

These results corroborate findings from Adebayo et al. [2018], Adebayo et al. [2020], and Nguyen et al. [2021] about how feature attribution methods generally struggle on debugging tasks.

**Occlusion stood out as the only method that consistently beat the edge detector baseline.**

Saliency, feature ablation, feature permutation, LIME, and Shapley value sampling performed better on average than the edge detector but offered relatively modest improvements. Occlusion [Zeiler and Fergus, 2014] consistently beat it. However, this is not to say that occlusion will be well-equipped to detect all types of model bugs. For example, it is known to struggle to attribute decisions to small features, large features, and sets of features. To

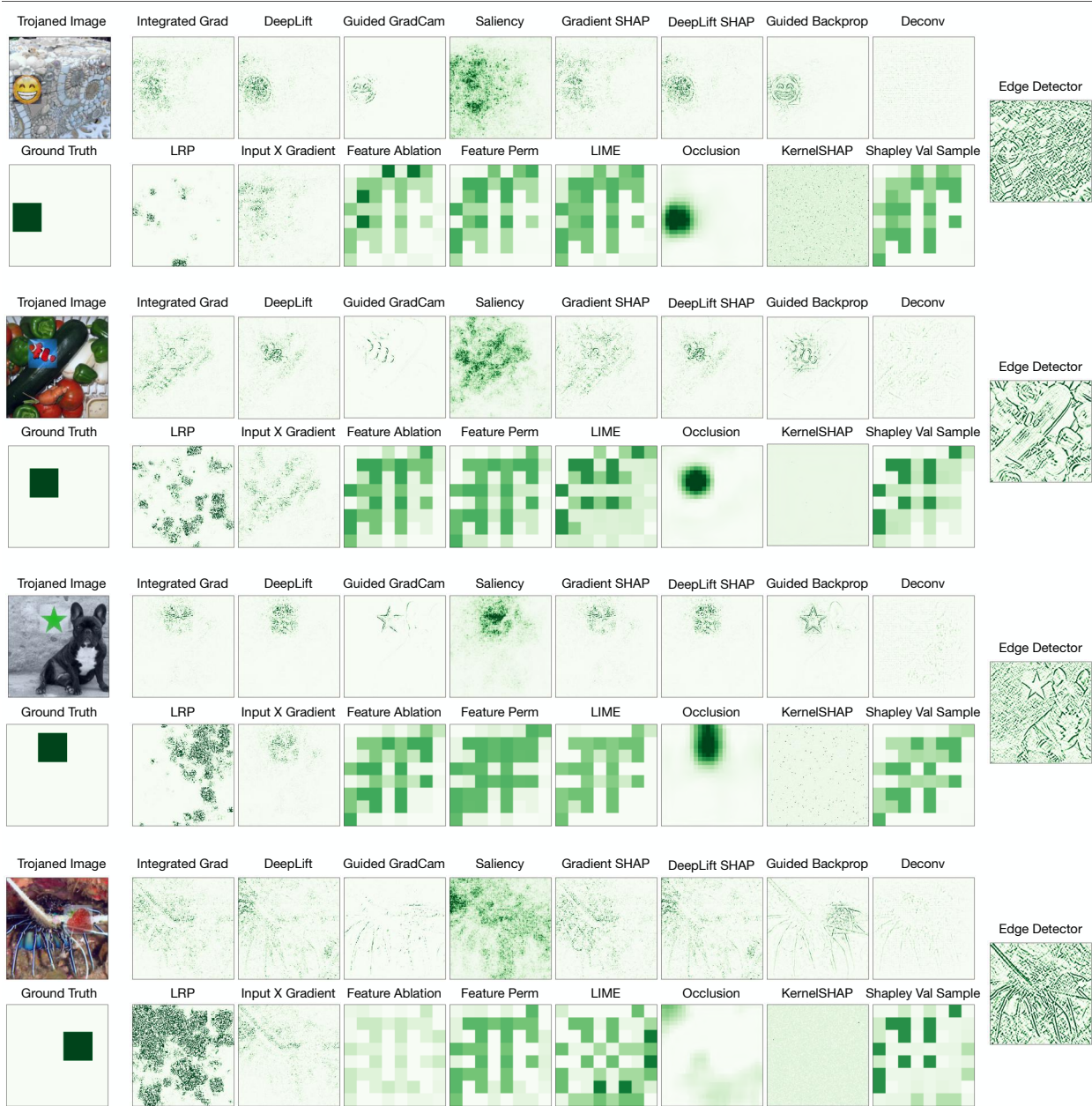


Figure 4.1: Examples of trojaned images, ground truth attribution maps, and attribution maps from various methods including an edge detector baseline. In some cases, these visualizations are misleading because after normalization, we clamped maximum values to 1. This clamping distorts differences between large values. See Figure 4.2 for quantitative results.

the best of our knowledge, no prior works on evaluating feature attribution/saliency with debugging tasks test occlusion (including Adebayo et al. [2018], Adebayo et al. [2020], and Nguyen et al. [2021]), so we cannot compare this finding to prior ones.

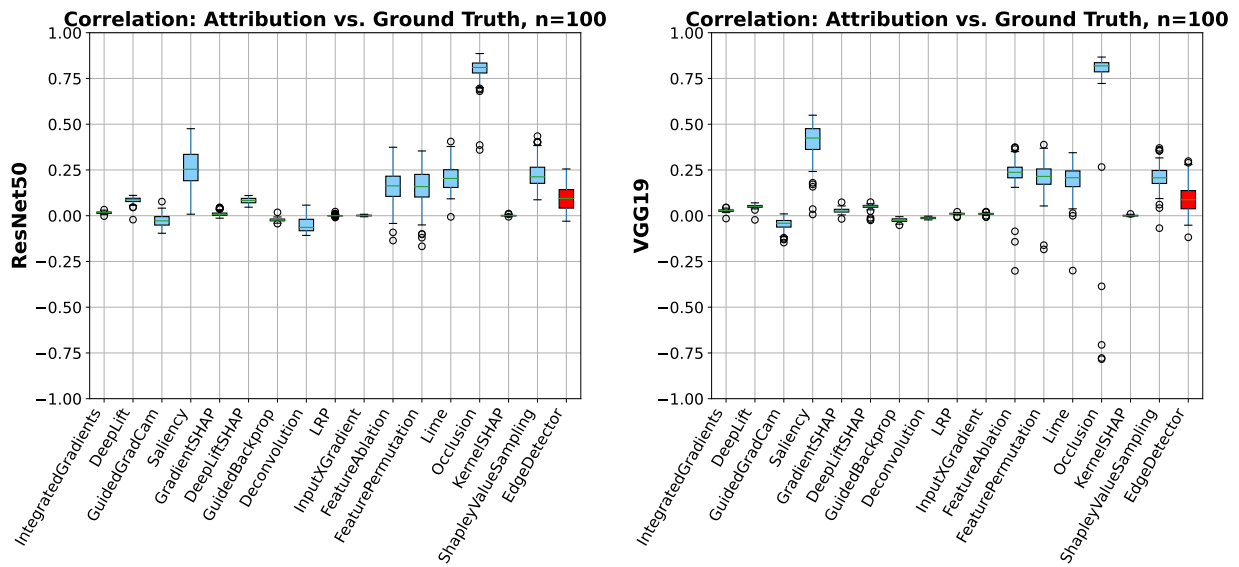


Figure 4.2: Correlations between attribution maps and ground truths for all 16 different feature attribution methods plus a simple edge detector when applied to a trojaned ResNet50 and VGG19. The edge detector baseline is shown in red. High values indicate better performance.

# Chapter 5

## Adversarial Feature Synthesis Methods Show Promise for Debugging

This chapter presents work from [Casper et al. \[2021b\]](#), [Casper et al. \[2022a\]](#), and [Casper et al. \[2023\]](#) done alongside coauthors Max Nadeau, Yuxiao Li, Tong Bu, Jiawei Li, Kevin Zhang, Kaivalya Hariharan, Gabriel Kreiman, and Dylan Hadfield-Menell.

The previous chapter demonstrated two challenges of feature attribution/saliency methods: (1) that they require data that triggers a model failure to help a human characterize that failure, and (2) that even under idealized conditions when access is given to data with features that trigger a failure, they still often struggle to highlight the trigger feature. These shortcomings motivate the use of other, dataset-free diagnostic tools. This chapter focuses on these.

In [Section 5.1](#) introduces a new technique based on human-describable adversarial attacks for interpreting networks. Then [Section 5.2](#) builds on this work by introducing a method to search for adversarial examples that are even easier for humans to describe because the adversarial features are restricted to natural objects. One thing that both of these techniques have in common is that they leverage model latents to synthesize/search for adversarial features. I also show how both can be used to discover bugs in models that would not be triggered by a typical test set. Finally, [5.3](#) benchmarks these methods along with others from prior work using the trojan-discovery task introduced in the previous chapter.

This chapter ultimately shows that feature synthesis methods do more with less than feature attribution/saliency ones. It also shows that the two methods introduced in this chapter perform better than similar prior techniques.

### 5.1 Robust Feature-Level Adversarial Attacks

This section presents work from [Casper et al. \[2021b\]](#) done alongside coauthors Max Nadeau, Gabriel Kreiman, and Dylan Hadfield-Menell.

Conventionally, adversarial inputs for visual classifiers take the form of small-norm perturbations to natural images [[Szegedy et al., 2013](#), [Goodfellow et al., 2014](#)]. These perturbations reliably cause confident misclassifications. However, to a human, they typically appear as

random or mildly-textured noise. Consequently, it is difficult to interpret these attacks—they rarely generalize to produce human-comprehensible insights about the target network. In other words, beyond the observation that such attacks are possible, it is hard to learn much about the underlying target network from these pixel-level perturbations.

In contrast, many real-world failures of biological vision are caused by perceptible, human-describable features. For instance, the [ringlet butterfly](#)’s predators are stunned by adversarial "eyespot" on its wings. This falls outside the scope of conventional adversarial examples because the misclassification results from a feature-level change to an object/image. The adversarial eyespots are robust in the sense that the same attack works across a variety of different observers, backgrounds, and viewing conditions. Furthermore, because the attack relies on high-level features, it is easy for a human to describe it.

This work takes inspiration from the ringlet butterfly’s eyespots and similar examples in which a model is fooled in the real world by an interpretable feature (e.g. [NTSB, 2018]). Our goal is to design adversaries that reveal easily-understandable weaknesses of the victim network. We focus on two desiderata for adversarial perturbations: attacks must be (1) interpretable (i.e. describable) to a human, and (2) robust so that interpretations generalize. We refer to these types of attacks as “feature-level” adversarial examples. Several previous works have created attacks by perturbing the latent representations of an image generator (e.g., Hu et al. [2021]), but thus far, approaches have been small in scale, limited in robustness, and not interpretability-driven.

We build on this prior work to propose an attack method that generates feature-level attacks against computer vision models. This method works on ImageNet scale models and creates robust, feature-level adversarial examples. We test three methods of introducing adversarial features into source images either by modifying the generator’s latents and/or inserting a generated patch into natural images. In contrast to previous works that have enforced the “adversarialness” of attacks only by inserting small features or restricting the distance between an adversary and a benign input, we also introduce methods that regularize the feature to be perceptible yet disguised to resemble something other than the target class.

We show that our method produces robust attacks that provide actionable insights into a network’s learned representations. Figure 5.1 demonstrates the interpretability benefits of this type of feature-level attack. It compares a conventional, pixel-level, adversarial patch, created using the method from Brown et al. [2017], with a feature-level attack using our method. While both attacks attempt to make a network misclassify a bee as a fly, the pixel-level attack exhibits high-frequency patterns and lacks visually-coherent objects. On the other hand, the feature-level attack displays easily describable features: the colored circles. We can validate this insight by considering the network performance when a picture of a traffic light is inserted into the image a bee. In this example, the image classification moves from a 55% confidence that the image is a bee to a 97% confidence that the image is of a fly. Section. 5.1.3 studies these types of “copy/paste” attacks more in depth.

This section’s contributions are threefold.

1. **Conceptual Insight:** We observe that robust feature-level adversaries can be used to produce useful types of inputs for studying the representations of deep networks
2. **Robust Attacks:** We introduce methods for generating feature-level adversaries that



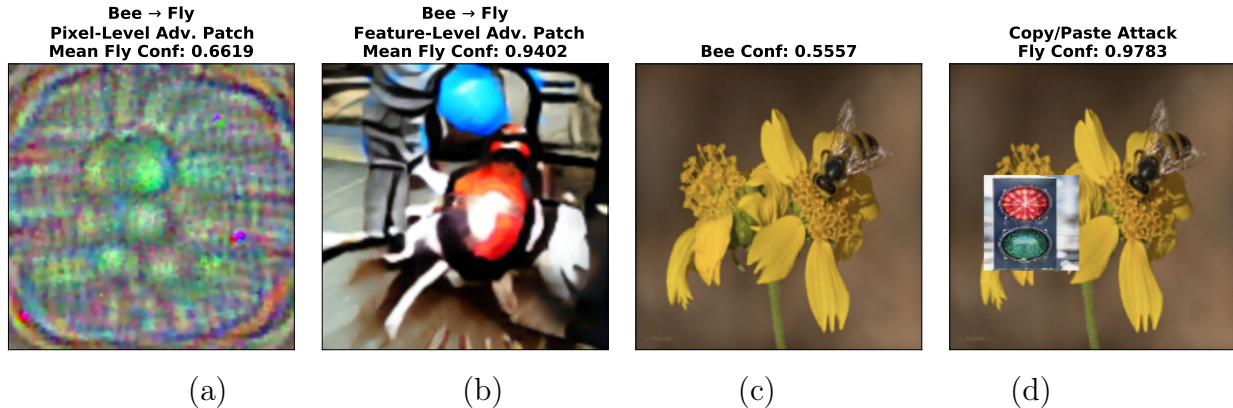


Figure 5.1: Our feature-level adversaries are useful for interpreting deep networks (we used a ResNet50 [He et al., 2016]). (a) A pixel-level adversarial patch trained to make images of bees misclassified as flies. (b) An analogous feature-level adversarial patch. (c) A correctly-classified image of a bee. (d) A successful copy/paste attack whose design was guided by adversarial examples like the one in (b).

are uniquely versatile and able to produce targeted, universal, disguised, physically-realizable, and black-box attacks at the ImageNet scale. See Table 5.1.

3. **Interpretability:** We generalize from our adversarial examples to design copy/paste attacks, verifying that our adversaries help us understand the network well enough to exploit it.

Code is available at [https://github.com/thestephencasper/feature\\_level\\_adv](https://github.com/thestephencasper/feature_level_adv).

### 5.1.1 Comparisons to Prior Work on Interpretable Adversarial Examples

Here, we contextualize our approach with others related to improving on conventional adversarial examples [Szegedy et al., 2013, Goodfellow et al., 2014]. Table 5.1 summarizes capabilities.

### 5.1.2 Methods

We adopt the “unrestricted” adversary paradigm [Song et al., 2018] under which an attack is successful if the network’s classification differs from an oracle’s (e.g., a human). Our adversaries can only change a small, fixed portion of either the generator’s latent or the image. We use white-box access to the network, though we present black-box attacks based on transfer from an ensemble in Appendix A.1.

Our attacks involve perturbing the latent representation in some layer of an image generator to produce an adversarial feature-level alteration. Figure 5.2 depicts our approach. We test three types of attacks, “patch”, “region” and “generalized patch” (plus a fourth in Casper et al. [2021b] which we call “channel” attacks). We find patch attacks to generally be the most successful.



	Targeted	Universal	Disguised	Physically-Realizable	Transferable/Black-Box	Copy/Paste	ImageNet Scale
Szegedy et al. [2013], Goodfellow et al. [2014]	✓	✗	✗	✗	✗	✗	✓
Natural mimics, e.g. Peacock, Ringlet Butterfly	✓	✓	✗	✓	✓	✗	N/A
Hayes and Danezis [2018]	✓	✓	✗	✗	✓	✗	✓
Mopuri et al. [2018a]	✓	✓	✗	✗	✓	✗	✓
Mopuri et al. [2018b]	✓	✓	✗	✗	✓	✗	✓
Poursaeed et al. [2018]	✓	✓	✗	✗	✓	✗	✓
Xiao et al. [2018]	✓	✗	✗	✗	✓	✗	✓
Hashemi et al. [2020]	✓	✓	✗	✗	✓	✗	✓
Wong and Kolter [2020]	✓	✗	✗	✗	✗	✗	✗
Liu et al. [2018]	✓	✗	✓	✗	✗	✗	✗
Samangouei et al. [2018]	✓	✗	✗	✗	✗	✗	✗
Song et al. [2018]	✓	✗	✓	✗	✓	✗	✗
Joshi et al. [2018]	✓	✗	✗	✗	✗	✗	✗
Joshi et al. [2019]	✓	✗	✓	✗	✗	✗	✗
Singla et al. [2019]	✓	✗	✗	✗	✓	✗	✗
Hu et al. [2021]	✓	✓	✓	✓	✗	✗	✗
Wang et al. [2020]	✓	✓	✓	✗	✗	✗	✗
Kurakin et al. [2016]	✓	✗	✗	✓	✓	✗	✓
Sharif et al. [2016]	✓	✗	✗	✓	✓	✗	✓
Brown et al. [2017]	✓	✓	✗	✓	✓	✗	✓
Eykholt et al. [2018]	✓	✗	✓	✓	✗	✗	✗
Athalye et al. [2018]	✓	✗	✗	✓	✗	✗	✓
Liu et al. [2019]	✓	✗	✗	✓	✓	✗	✓
Thys et al. [2019]	✓	✓	✗	✓	✗	✗	✗
Kong et al. [2020]	✓	✗	✗	✓	✗	✗	✗
Komkov and Petiushko [2021]	✓	✓	✗	✓	✗	✗	✗
Dong et al. [2017]	✓	✗	✗	✗	✓	✗	✓
Geirhos et al. [2018]	✗	✗	✗	✗	✗	✗	✓
Leclerc et al. [2021]	✗	✗	✓	✗	✗	✓	✓
Wiles et al. [2022]	✗	✗	✓	✗	✓	✗	✓
Carter et al. [2019]	✓	✗	✓	✗	✗	✓	✓
Mu and Andreas [2020]	✗	✗	✓	✗	✗	✓	✓
Hernandez et al. [2022]	✗	✗	✓	✗	✗	✓	✓
<b>Ours</b>	✓	✓	✓	✓	✓	✓	✓

Table 5.1: Our feature-level attacks are uniquely versatile. Each row represents a related work (in the order in which they are presented in Chapter 2.) Each column indicates a demonstrated capability of a method. Note that two methods each having a ✓ for a capability does not imply they do so equally well. *Targeted*=working for an arbitrary target class. *Universal*=working for any source example. *Disguised*=Perceptible and resembling something other than the target class. *Physically-realizable*=working in the physical world. *Transferable/black-box*=transferring to other classifiers. *Copy/Paste*=useful for designing attacks in which a natural feature is pasted into a natural image.

**Patch:** We use the generator to produce a square patch that is inserted into a natural image [Sharma et al., 2022].

**Region:** Starting with some generated image, we randomly select a square column of the latent in a generator layer which spans the channel dimension and replace it with a learned insertion. This is analogous to a square patch in the pixel representation. We keep insertion location fixed over training. The modified latent is passed through the rest of the generator, producing the adversarial image.

**Generalized Patch:** These patches can be of any shape, hence the name “generalized” patch. We first generate a region attack and then extract a generalized patch from it. We do

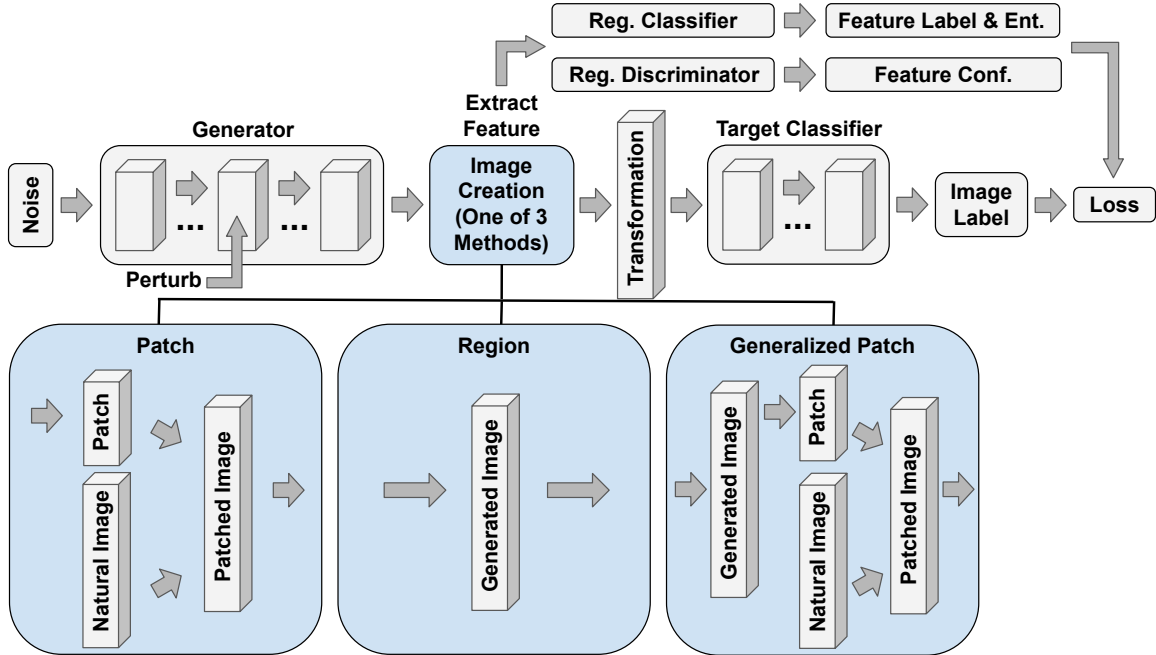


Figure 5.2: Our fully-differentiable pipeline for creating feature-level attacks. In each experiment, we create either “patch,” “region,” or “generalized patch” attacks. The regularization terms in the loss based on an external classifier and discriminator are optional and are meant to make the inserted feature appear disguised as some non-target class.

this by taking the absolute-valued pixel-level difference between the original and adversarial image, applying a Gaussian filter for smoothing, and creating a binary mask from the top decile of these pixel differences. We apply this mask to the generated image to isolate the region that the perturbation altered. We can then treat this as a patch and overlay it onto an image in any location.

## Basic Attacks

For all attacks, we train a perturbation  $\delta$  to the latent of the generator to minimize a loss that optimizes for both attacking the classifier and appearing interpretable:

$$\arg \min_{\delta} \mathbb{E}_{x \sim \mathcal{X}, t \sim \mathcal{T}, l \sim \mathcal{L}} L_{\text{x-ent}}[C(A(x, \delta, t, l)), y_{\text{targ}}] + L_{\text{reg}}[A(x, \delta, t, l)] \quad (5.1)$$

with  $\mathcal{X}$  a distribution over source images (e.g., a dataset or generation distribution),  $\mathcal{T}$  a distribution over transformations,  $\mathcal{L}$  a distribution over insertion locations (this only applies for patches and generalized patches),  $C$  the target classifier,  $A$  an image-generating function,  $L_{\text{x-ent}}$  a targeted crossentropy loss for attacking the classifier,  $y_{\text{targ}}$  the target class, and  $L_{\text{reg}}$  a regularization loss. The adversary has no control over  $\mathcal{X}$ ,  $\mathcal{T}$ , or  $\mathcal{L}$ , so it must learn features that work on the network independent of any particular source image, transformation, or insertion location. For all of our attacks,  $L_{\text{reg}}$  contains a total variation loss,  $TV(a)$ , to discourage high-frequency patterns.

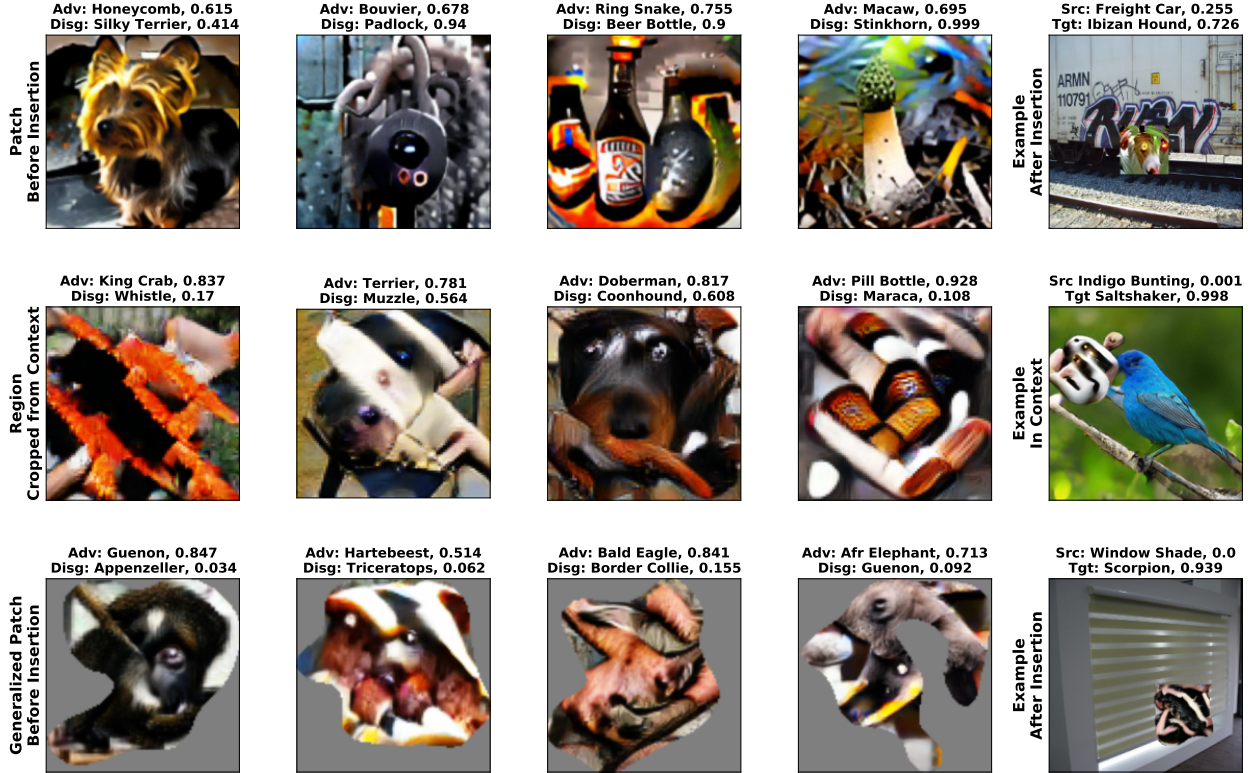


Figure 5.3: Examples of targeted, universal feature-level adversaries from patch (top), region (middle), and generalized patch (bottom) attacks. The first four columns show the adversarial features. The mean target class confidence is labeled ‘Adv.’ and is calculated under random source images (and random insertion locations for patch and generalized patch attacks). The target network’s disguise class confidence for each patch or extracted generalized patch is labeled ‘Disg.’ The final column shows examples of the features applied to images. The example image for each is labeled with its source and target class confidences.

### “Disguised” Attacks

Ideally, a feature-level adversarial example should appear to a human as easily-describable but should not resemble the attack’s target class. We call such attacks “disguised.” Here, the main goal is not to fool a human, but to help them *learn* about what types of realistic features might cause the model to make a mistake. To train these disguised attacks, we use additional terms in  $L_{\text{reg}}$  as proxies for these two criteria. We differentially resize the patch or the extracted generalized patch and pass it through a GAN discriminator and auxiliary classifier. We then add weighted terms to the regularization loss based on the discriminator’s ( $D$ ) logistic loss for classifying the input as fake, the output entropy ( $H$ ) of some classifier ( $C'$ ), and/or the negative of the classifier’s crossentropy loss for labeling the input as the attack’s target class. Note that  $C'$  could either be the same or different than the target classifier  $C$ . With all of these terms, the regularization objective is

$$L_{\text{reg}}(a) = \lambda_1 TV(a) + \underbrace{\lambda_2 L_{\text{logistic}}[D(P(a))] + \lambda_3 H[C'(P(a))] - \lambda_4 L_{\text{x-ent}}[C'(P(a), y_{\text{targ}})]}_{\text{“Disguise” Regularizers}}. \quad (5.2)$$

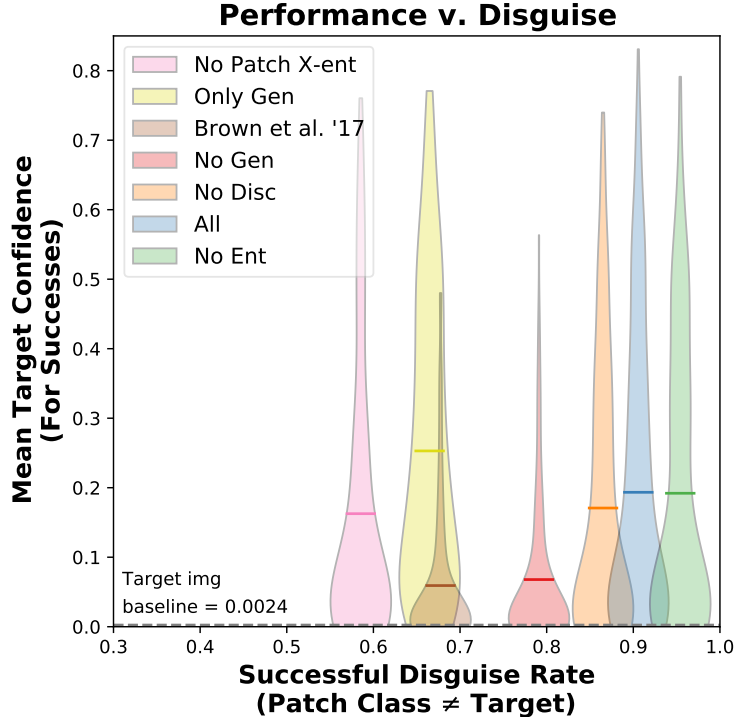


Figure 5.4: Targeted, universal patch attacks compared. Successful disguise success rate (x axis) shows the proportion of attacks in which the patch was not classified by the network as the target class when viewed on its own. Mean target class confidence (y axis) gives the empirical target class confidences of 250 patch attacks. Each is an average over 100 source images. The proportion of each distribution above 0.5 gives a lower-bound for the top-1 attack success rate. The mean target class confidence for using randomly-sampled natural target class images as patches is 0.0024 and is shown as a thin dotted line at the bottom.

Here,  $P(a)$  returns the extracted and resized patch from adversarial image  $a$ . In order, these three new terms encourage the adversarial feature to (1) look realistic, and (2) look like some specific class, but (3) not the target class. The choice of disguise class is left entirely to the training process.

### 5.1.3 Experiments

We use BigGAN generators from [Brock et al., 2018, Wolf, 2018], and perturb the post-ReLU outputs of the internal ‘GenBlocks.’ We also found that training slight perturbations to the BigGAN’s inputs improved performance. We used the BigGAN discriminator and adversarially trained classifiers from Engstrom et al. [2019a] for disguise regularization. By default, we attacked a ResNet50 [He et al., 2016], restricting patch attacks to 1/16 of the image and region and generalized patch attacks to 1/8. First, in Section 5.1.3 we show that these feature-level adversaries are highly robust to suggest that interpretations based on them are generalizable. Second, in Section 5.1.3 we put these interpretations to the test and show that our feature-level adversaries can help one understand a network well enough

to exploit it.

## Robust Attacks

Figure 5.3 shows examples of targeted, universal, and disguised feature-level patch (top), region (middle), and generalized patch (bottom) attacks which were each trained with all of the disguise regularization terms from Eq. 5.2. We find the disguises to be effective, particularly for the patches (top row), but imperfect.

**Performance versus Disguise:** Here, we study our patch attacks in depth to test how effective they are at attacking the network and how successfully they can help to identify non-target-class features that can fool the network. We compared seven different approaches. The first was our full approach using the generator and all disguise regularization terms from Eq. 5.2. The rest were ablation tests in which we omitted the generator (No Gen), the discriminator (No Disc) regularization term (No Reg), the entropy regularization term (No Ent), the crossentropy regularization term (No Patch X-ent), all three regularization terms (Only Gen), and finally the discriminator and all three regularization terms Brown et al. [2017]. This final unregularized, pixel-level method resulted in the same approach as Brown et al. [2017]. For each test, all else was kept identical including penalizing total variation, training under transformations, and initializing the patch as a generator output.

For each method, we generated universal attacks with random target classes until we obtained 250 successfully “disguised” ones in which the resulting adversarial feature was not classified by the network as the target class when viewed on its own. Figure 5.4 plots the success rate versus the distribution of target class mean confidences for each type of attack. For all methods, these universal attacks have variable target class confidences due in large part to the random selection of target class. Some attacks are stochastically Pareto dominated by others. For example, the pixel-space Brown et al. [2017] attacks were the least effective at attacking the target network and had the third least disguise rate. In other cases, there is a tradeoff between attack performance and disguise which can be controlled using the regularization terms from Eq. 5.2. We also compare our attacks to two baselines using resized natural images from the target class and randomly sampled patches from the center of target class images. These resulted in a mean target class confidences of 0.0024 and 0.0018 respectively.

Notably, Figure 5.4 does not capture everything that one might care about in these attacks. It does not show any measure of how “realistic” the resulting patches look.

In Section 5.1.3, Figure 5.4 plots the successful disguise rate of attacks alongside their distribution of mean target class confidences. However, this leaves out how effective each type of attack is at appearing realistic to a human. Figure 5.5 aims to measure this by using the target class confidence of an Inception-v3 Szegedy et al. [2016] as a proxy for how realistic a patch appears to a human. Figure 5.5 plots the mean target class confidences for successfully disguised attacks versus their Inception-v3 disguise label confidence. This suggests that the attacks that are the best at producing realistic-looking patches are the “All” ones with the generator and all regularization terms and the “No Disc” ablations which omit the discriminator regularization term.

?? shows the same target class confidence data from the  $y$  axis in Figure 5.4 versus the disguise class label confidence from an Inception-v3 which we use as a proxy for how realistic

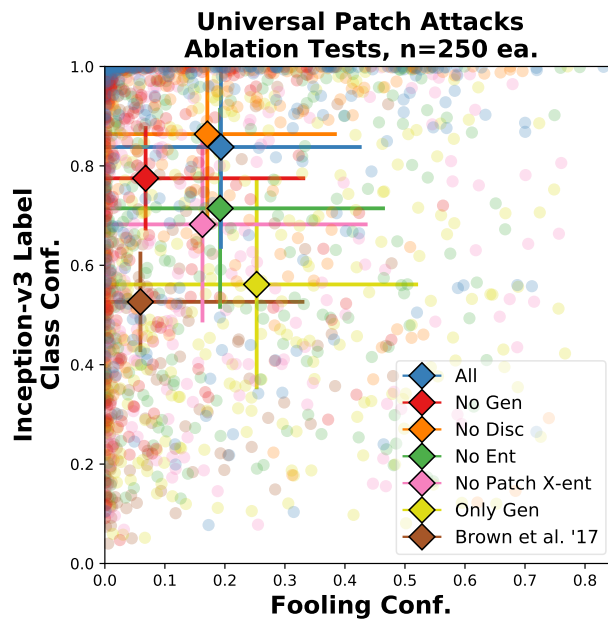


Figure 5.5: Targeted, universal patch attacks compared by mean target class confidence and Inception-v3 label-class confidence. Inception-v3 class conf. on the  $x$ -axis gives the mean target class confidence from the attacked network for images which have the patch inserted. The Inception-v3’s label class confidence for the patch on the  $y$ -axis is used as a proxy for human interpretability. Attacks further up and right are better. Centroids are shown with error bars giving the standard deviation.



a human would find the patch. It suggests that the best attacks for producing patches that appear realistic are the “All” and “No Disc” methods. Because they were initialized from generator outputs, some of the Brown et al. [2017] attacks have a veneer-like resemblance to non-target class features. Nonetheless, they contain higher-frequency patterns and less coherent objects in comparison to the two sets of feature-level attacks. We subjectively find the “All” attacks to be the best disguised.



Figure 5.6: Examples of targeted, disguised, universal, and physically-realizable feature-level attacks. See Casper et al. [2021b] for full-sized versions of the patches.

**Physical-Realizability:** To test their ability to transfer to the physical world, we generated 100 additional targeted, universal, and disguised adversarial patches. We used the generator and all regularization terms (the “All” condition from above). We selected the 10 with the best mean target class confidence, printed them, and photographed each next to 9 objects from different ImageNet classes.<sup>1</sup> We confirmed that photographs of each object were correctly classified without a patch. Figure 5.6 shows successful examples. The mean and standard deviation of the target class confidences for our attacks in the physical world were 0.312 and 0.318 respectively ( $n = 90$ , not i.i.d.). This means that these patches’ mean effectiveness dropped by less than  $\frac{1}{2}$  when transferring to the physical world.

**Black-Box Attacks:** In Appendix A.1, we show that our targeted universal attacks can transfer from an ensemble to a held-out model.

## Interpretability

If an adversarial feature successfully fools the victim network, this suggests that the network associates that feature in the context of a source image with the target class. These adversaries can suggest both beneficial and harmful feature-class associations.

Simply developing an interpretation, however, is easy. Showing that one leads to a useful understanding of the network is harder. One challenge in the explainable AI literature is to develop interpretations that go beyond seeming plausible and stand up to scrutiny [Räuber et al., 2022]. Robust feature-level adversarial patches can easily be used to develop hypotheses about the network’s behavior, e.g. “The network thinks that bee features plus colorful balls implies a fly.” But are these valid, useful interpretations of the network? In other words, are our adversaries adversarial because of their interpretable qualities, or is it because of hidden motifs? We verify interpretations by using our attacks to make and validate predictions about how to fool the target network with natural objects.

<sup>1</sup>Backpack, banana, bath towel, lemon, jeans, spatula, sunglasses, toilet tissue, and toaster.

**Validating Interpretations with Copy/Paste Attacks** A “copy-paste” attack is created by inserting one natural image into another to cause an unexpected misclassification. They are more restricted than patch attacks because the features pasted into an image must be natural objects. As a result, they are of high interest for physically realizable attacks because they suggest combinations of real objects that yield unexpected classifications. They also have precedent in the real world. For example, subimage insertions into pornographic images have been used to evade NSFW content detectors [Yuan et al., 2019].

<p><b>Bee → Fly Traffic Light?</b></p>	
<p><b>Orig Mean Conf: 0.2982 Adv Mean Conf: 0.8661</b></p>	
<p><b>Traffic Light → Fly Bee/Wings?</b></p>	
<p><b>Orig Mean Conf: 0.0 Adv Mean Conf: 0.2148</b></p>	
<p><b>Indian → Afr. Elephant Blue Coloration?</b></p>	
<p><b>Orig Mean Conf: 0.4368 Adv Mean Conf: 0.7263</b></p>	
<p><b>Puffer → Lionfish Butterfly Wings?</b></p>	
<p><b>Orig Mean Conf: 0.0195 Adv Mean Conf: 0.4645</b></p>	



To develop copy/paste attacks, we select a source and target class, generate class-universal adversarial features, and manually analyze them for motifs that resemble natural objects. Here, we used basic attacks without the disguise regularization terms from Eq. 5.2. We then paste images of these objects into natural images and pass them through the classifier.

Section 5.1.3 shows four types of copy/paste attacks. In each odd row, we show six patch, region, and generalized patch adversaries that were used to guide the design of a copy/paste attack. In each even row are the copy/paste adversaries for the 6 (of 50) images for the source class for which the insertion resulted in the highest target class confidence increase along with the mean target class confidences before and after patch insertion for those 6. The success of these attacks shows their usefulness for interpreting the target network because they require that a human understands the mistake the model is making like “Bee  $\wedge$  Traffic Light  $\rightarrow$  Fly” well enough to manually exploit it. Given the differences in the adversarial features that are produced in the Bee  $\rightarrow$  Fly and Traffic Light  $\rightarrow$  Fly attacks, Section 5.1.3 also demonstrates how our attacks take the distribution of source images into account.

**Comparisons to Other Methods:** Three prior works [Carter et al., 2019, Mu and Andreas, 2020, Hernandez et al., 2022] have developed copy/paste attacks, also via interpretability tools. Unlike Mu and Andreas [2020], Hernandez et al. [2022], our approach allows for targeted attacks. And unlike all three, rather than simply identifying features associated with a class, our adversaries generate adversarial features for a target class *conditional* on any distribution over source images (i.e. the source class) with which the adversaries are trained. Little work has been done on copy/paste adversaries, and thus far, methods have either not allowed for targeted attacks or have required a human in the loop. This makes objective comparisons difficult. However, we provide examples of a feature visualization-based method inspired by Carter et al. [2019] in Appendix A.2 to compare with ours. See Appendix A.2 for visualizations and discussion. In short, for the Indian  $\rightarrow$  African Elephant attack, the source and target class share many features, and we find no evidence that feature visualization is able to suggest useful features for copy/paste attacks. This suggests that our attacks’ ability to take the source image distribution into account may be more helpful for discovering certain weaknesses compared to the baseline inspired by Carter et al. [2019].

## 5.1.4 Conclusion and Broader Impacts for Robust Feature-Level Adversaries

### Contributions

Here we use feature-level adversarial examples to attack and interpret deep networks in order to contribute to a more practical understanding of network vulnerabilities. As an attack method, our approach is versatile. It can produce targeted, universal, disguised, physically realizable, black-box, and copy/paste attacks at the ImageNet scale. This method can be also used as an interpretability tool to help diagnose flaws in models. We ground the notion of interpretability in the ability to make predictions about combinations of natural features that will make a model fail. And finally, we demonstrate this through the design of targeted copy/paste attacks for any distribution over source inputs.

## Implications

Like any work on adversarial attacks, our approach could be used maliciously to make a system fail, but we emphasize their diagnostic value. Understanding threats is a prerequisite to avoiding them. Given the robustness and versatility of our attacks, we argue that they may be valuable for continued work to address threats that systems may face in practical applications. There are at least two ways in which these methods can be useful.

## Adversarial Training

The first is for adversarial training. Training networks on adversarial images has been shown to improve their robustness to the attacks that are used [Engstrom et al., 2019a]. But this does not guarantee robustness to other types of adversarial inputs (e.g. [Hendrycks et al., 2021b]). Our feature-level attacks are categorically different from conventional pixel-level ones, and our copy/paste attacks show how networks can be fooled by novel combinations of natural objects, failures that are outside the conventional paradigm for adversarial robustness (e.g., Engstrom et al. [2019a]). Consequently, we expect that adversarial training on broader classes of attacks such as the one we propose here will be valuable for designing more robust models.

## Diagnostics

The second is for rigorously diagnosing flaws. We show that feature-level adversaries aid the discovery of exploitable spurious feature/class associations (Section 5.1.3). Our approach could also be extended beyond what we have demonstrated here. For example, our methods may be useful for feature visualization [Olah et al., 2017] of a network’s internal neurons. An analogous approach to ours can also be used in Natural Language Processing [Song et al., 2020, Perez et al., 2022]. Furthermore, it may be valuable to use these adversaries to identify generalizable flaws in networks that humans can easily understand but with minimal human involvement.

## Limitations

A limitation of our approach is that when multiple desiderata are optimized for at the same time (e.g., universality + transformation robustness + disguise), attacks are generally less successful, more time-consuming, and require more screening to find good ones. This could be a bottleneck for large-scale adversarial training. Ultimately, this type of attack is limited by the efficiency and quality of the generator, so future work should leverage advances in generative modeling. Our evaluation method is also limited to a proof-of-concept for the design of copy/paste attacks. Future work should evaluate this more rigorously. We are currently working toward developing a benchmark for interpretability tools based on their ability to aid a human in rediscovering trojans [Geigel, 2013] that have been implanted into a model.

## Conclusion

Each of the 11 proposals for building safe AI outlined in Hubinger [2020] explicitly calls for adversarial robustness and/or interpretability tools, and recent work from Ziegler et al. [2022] on high-stakes reliability in AI found that interpretability tools strengthened their ability to produce inputs for adversarial training. Given the close relationship between interpretability and adversarial robustness, continued study of the connections between them will be key for building safer AI systems.

## 5.2 Search for Natural Adversarial Features Using Embeddings

This section presents work from Casper et al. [2022a] and Casper et al. [2023] done alongside coauthors Yuxiao Li, Tong Bu, Jiawei Li, Kevin Zhang, Keivalya Hariharan, and Dylan Hadfield-Menell.

As discussed in the previous section, robust feature-level adversarial examples can help in generating practical interpretations of how networks operate. Typically, adversarial examples are generated by optimizing perturbations to the input of a network. Some previous works offer examples of adversaries being used to develop generalizable interpretations of DNNs Dong et al. [2017], Tomsett et al. [2018], Ilyas et al. [2019], Casper et al. [2022b].

However, there are limitations to what one can learn about flaws in DNNs from synthesized features Borowski et al. [2020]. First, synthetic adversarial perturbations are often difficult to describe and thus offer limited help with human-centered approaches to interpretability. Second, even when synthetic adversarial features are interpretable, it is unclear without additional testing whether they fool a DNN due to their interpretable features or due to hidden motifs Brown et al. [2017], Ilyas et al. [2019]. This makes developing generalizable understandings with them difficult. Third, there is a gap between research and practice in adversarial robustness Apruzzese et al. [2022]. Real-world failures of DNNs are often due to atypical natural features or combinations thereof Hendrycks et al. [2021b], but synthesized features are off this distribution.

Here, I present work from Casper et al. [2023] to diagnose weaknesses in DNNs using *natural, interpretable* features in a way that builds off of the technique from the previous section. We introduce using a Search for Natural Adversarial Features Using Embeddings (SNAFUE) to find novel adversarial combinations of natural features. We apply SNAFUE to find *copy/paste* attacks for an image classifier in which one natural image is inserted as a patch into another to induce a targeted misclassification. Because these attacks are based on novel combinations of natural features, they are particularly easy for humans to describe and learn from. Figure 5.16 outlines this approach. First, we use a generator to synthesize robust feature-level adversarial patches Casper et al. [2022b] which are designed to make any image from a particular source class misclassified as a target. Second, we use the target model’s latent activations to create embeddings of both these synthetic patches and a dataset of natural patches. Finally we select the natural patches that embed most similarly to the synthetic ones.

We apply SNAFUE at the ImageNet scale. First, we use SNAFUE to replicate all successful known examples of copy/paste attacks from previous works with no human involvement. Second, we demonstrate its scalability by identifying hundreds of vulnerabilities. Figure 5.10 and Figure 5.9 show examples that illustrate easily describable misassociations between features and classes in the network. Overall, this work with SNAFUE offers two key contributions.

1. **Algorithmic:** We introduce Search for Natural Adversarial Features Using Embeddings (SNAFUE) as a tool for scalable human oversight.
2. **Diagnostic:** We apply SNAFUE by red-teaming an image classifier. We demonstrate that it automatically identifies weaknesses due to natural features that are uniquely human-interpretable.

The key advantages of SNAFUE involve scalably generating adversarial data that is naturally interpretable due to the fact that they are composed of combinations of natural features. This makes the adversarial examples generated by SNAFUE uniquely interpretable. As will be shown in the next section, this sets SNAFUE apart distinctly from other comparable tools for interpreting networks via attacks. Code for SNAFUE is available at <https://github.com/thestephencasper/snafue>.

### 5.2.1 Methodology

For all experiments here with SNAFUE, we report the *success rate* defined as the proportion of the time that a patched image was classified as the target class minus the proportion of the time the unpatched natural image was.

#### Robust feature-level adversarial patches:

First, we create synthetic robust feature-level adversarial patches as in Casper et al. [2022b] by perturbing the latent activations of a BigGAN Brock et al. [2018] generator. Unlike Casper et al. [2022b], we do not use a GAN discriminator for regularization or use an auxiliary classifier to regularize for realistic-looking patches. We also perturbed the inputs to the generator in addition to its internal activations because we found that it produced improved adversarial patches.

#### Candidate patches:

Patches for SNAFUE can come from any source and do not need labels. Features do not necessarily have to be natural and could, for example, be procedurally generated. Here, we used a total of  $N = 265,457$  natural images from five sources: the ImageNet validation set Russakovsky et al. [2015] (50,000) TinyImageNet Le and Yang [2015] (100,000), OpenSurfaces Bell et al. [2013] (57,500), the non OpenSurfaces images from Broden Bau et al. [2017a] (37,953).

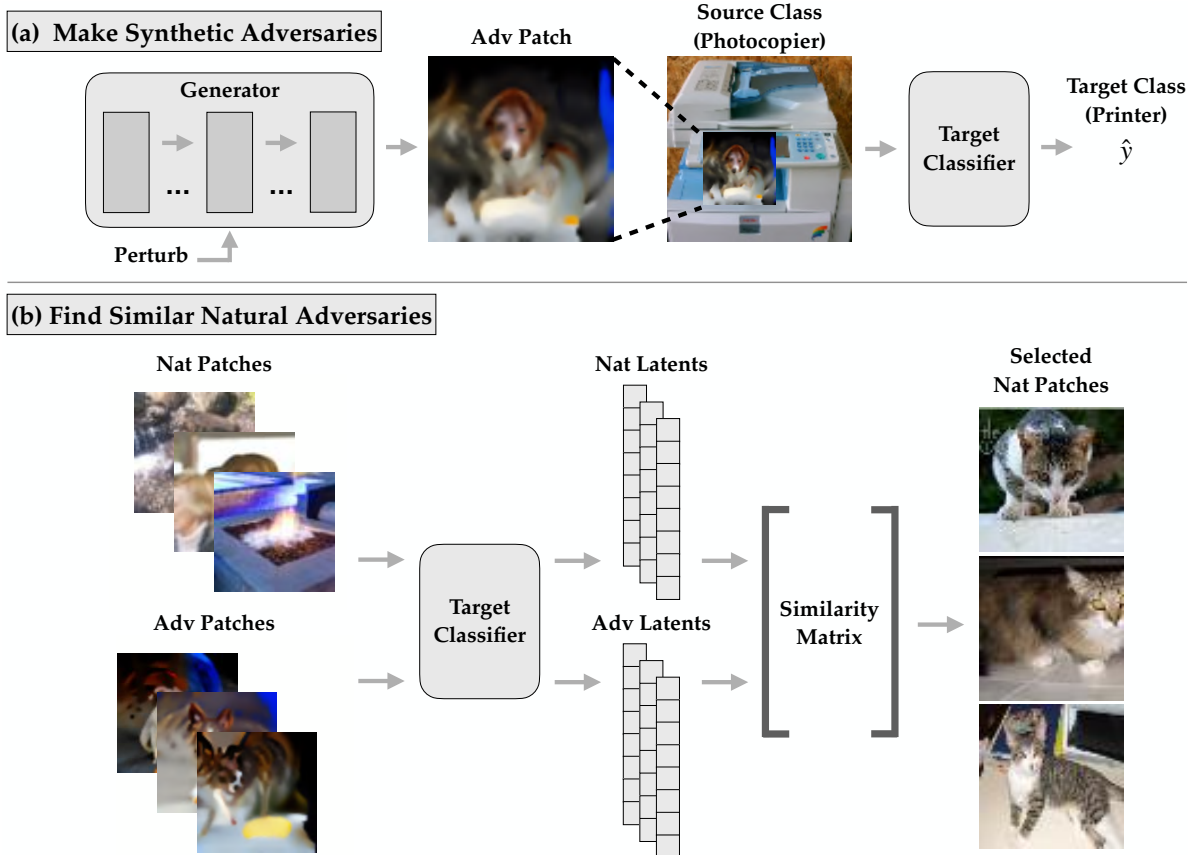


Figure 5.7: SNAFUE, our automated method for finding targeted adversarial combinations of natural features. This example illustrates an experiment that found that cats can make photocopiers misclassified as printers. (a) First, we create feature-level adversarial patches as in Casper et al. [2022b] by perturbing the latent activations of a generator. (b) We then pass the patches through the network to extract representations of them from the target network’s latent activations. Finally, we select the natural patches whose latents are the most similar to the adversarial ones.

### Image and patch scaling:

All synthetic patches were parameterized as  $64 \times 64$  images. Each was trained under transformations including random resizing. Similarly, all natural patches were  $64 \times 64$  pixels. All adversarial patches were tested by resizing them to  $100 \times 100$  and inserting them into  $256 \times 256$  source images at random locations.

### Embeddings:

We used the  $N = 265,457$  natural patches along with  $M = 10$  adversarial patches and passed them through the target network to get an  $L$ -dimensional embedding of each using the post-ReLU latents from the penultimate (avgpooling) layer of the target network (which we found to be more effective than other embedding methods). The result was a nonnegative  $N \times L$  matrix  $U$  of natural patch embeddings and a  $M \times L$  matrix  $V$  of adversarial patch

embeddings. A different  $V$  must be computed for each attack, but  $U$  only needs to be computed once. This plus the fact that embedding the natural patches does not require insertion into a set of source images makes SNAFUE much more efficient than a brute-force search. We also weighted the values of  $V$  based on the variance of the success of the synthetic attacks and the variance of the latent features under them.

**Weighting:**

To reduce the influence of embedding features that vary widely across the adversarial patches, we apply an  $L$ -dimensional elementwise mask  $w$  to the embedding in each row of  $V$  with weights

$$w_j = \begin{cases} 0 & \text{if } \text{cv}_i(V_{ij}) > 1 \\ 1 - \text{cv}_i(V_{ij}) & \text{else} \end{cases}$$

where  $\text{cv}_i(V_{ij})$  is the coefficient of variation over the  $j$ 'th column of  $V$ , with  $\mu_j = \frac{1}{M} \sum_i V_{ij} \geq 0$  and  $\text{cv}_i(V_{ij}) = \frac{\sqrt{\frac{1}{M-1} \sum_i (V_{ij} - \mu_j)^2}}{\mu_j + \epsilon}$  for some small positive  $\epsilon$ .

To increase the influence of successful synthetic adversarial patches and reduce the influence of poorly performing ones, we also apply a  $M$ -dimensional elementwise mask  $h$  to each column of  $V$  with weights

$$h_i = \frac{\delta_i - \delta_{\min}}{\delta_{\max} - \delta_{\min}}$$

where  $\delta_i$  is the mean fooling confidence increase of the post-softmax value of the target output neuron under the patch insertions for the  $i^{\text{th}}$  synthetic adversary. If any  $\delta$  is negative, we replace it with zero, and if the denominator is zero, we set  $h_i$  to zero.

Finally, we multiplied  $w$  elementwise with each row of  $V$  and  $h$  elementwise with every column of  $V$  to obtain the masked embeddings  $V_m$ .

**Selecting natural patches:**

We then obtained the  $N \times M$  matrix  $S$  of cosine similarities between  $U$  and  $V$ . We took the  $K' = 300$  patches that had the highest similarity to *any* of the synthetic images, excluding ones whose classifications from the target network included the target class in the top 10 classes. Finally, we evaluated all  $K'$  natural patches under random insertion locations over all 50 source images from the validation set and subsampled the  $K = 10$  natural patches that increased the target network’s post-softmax confidence in the target class the most. Screening the  $K'$  natural patches for the best 10 caused only a marginal increase in computational overhead. The method was mainly bottlenecked by the cost of training the synthetic adversarial patches (for 64 batches of 32 insertions each). The numbers of screened and selected patches are arbitrary, and because it is fully automated, SNAFUE allows for flexibility in how many synthetic adversaries to create and how many natural adversaries to screen. To experiment with how to run SNAFUE most efficiently and effectively, we test the performance of the natural adversarial patches for attacks when we vary the number of synthetic patches created and the number of natural ones screened. We did this for 100

randomly sampled pairs of source and target classes and evaluated the top 10. Figure 5.8 shows the results.

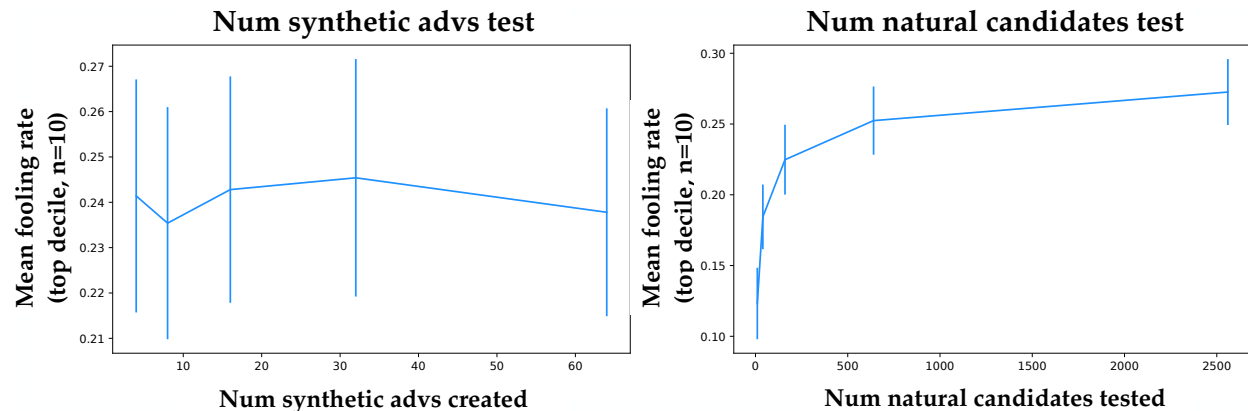


Figure 5.8: (Left) Mean natural patch success rate as a function of the number of synthetic adversaries we created, from which we selected the best 10 (or took all if there were fewer than 10) to then use in the search for natural patches. (Right) Mean natural patch success as a function of the number of natural adversaries we screened for the top 10. Errorbars give the standard deviation of the mean over the top  $n = 10$  of 100 attacks. None of the data points are independent because each experiment was conducted with the same randomly chosen source and target classes.

## 5.2.2 Examples:

We provide additional examples of copy/paste attack patches from SNAFUE in Figure 5.9. We present additional examples in Figure 5.10 and argue that SNAFUE can be used to discover distinct types of flaws.

## 5.2.3 Experiments

### Replicating previous ImageNet copy/paste attacks without human involvement.

First, we set out to replicate *all* known successful ImageNet copy/paste attacks from previous works without any human involvement. To our knowledge, there are 9 such attacks, 3 each from Carter et al. [2019], Hernandez et al. [2022]<sup>2</sup> and Section 5.1.<sup>3,4</sup> We used SNAFUE to find 10 natural patches for all 9 attacks. Figure 5.11 shows the results. In all cases, we are able to find successful natural adversarial patches. In most cases, we find similar adversarial

<sup>2</sup>The attacks presented in Hernandez et al. [2022] were not universal within a source class and were only developed for a single source image each. When replicating their results, we use the same single sources. When replicating attacks from the other two works, we train and test the attacks as source class-universal ones.

<sup>3</sup>Section 5.1 tests a fourth attack involving patches making traffic lights appear as flies, the examples they identified were not successful at causing targeted misclassification.

<sup>4</sup>Mu and Andreas [2020] also test copy paste attacks, but not on ImageNet networks





Figure 5.9: Examples of natural adversarial patches for several targeted attacks. Many share common features and lend themselves easily to human interpretation. Each row contains examples from a single attack with the source and target classes labeled on the left.



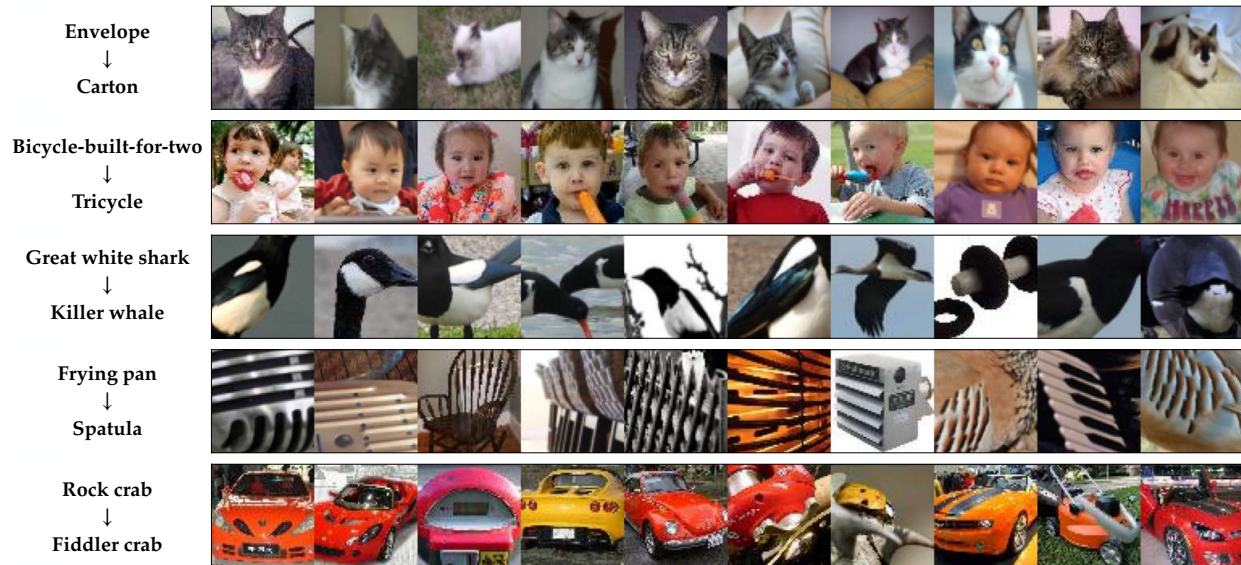


Figure 5.10: SNAFUE identifies distinct *types* of problems. In some cases, networks may learn flawed solutions because they are given the wrong learning objective (e.g. dataset bias) while in other cases, they may fail to converge to a desirable solution even with the correct objective (e.g. misgeneralization). SNAFUE can discover both types of issues. In some cases, it discovers failures that result from dataset biases. Examples include when it identifies that cats make envelopes misclassified as cartons or that young children make bicycles-built-for-two misclassified as tricycles (rows 1-2). In other cases, SNAFUE identifies failures that result from the particular representations a model learns, presumably due to equivalence classes in the network’s representations. Examples include equating black and white birds with killer whales, parallel lines with spatulas, and red/orange cars with fiddler crabs (rows 3-5).

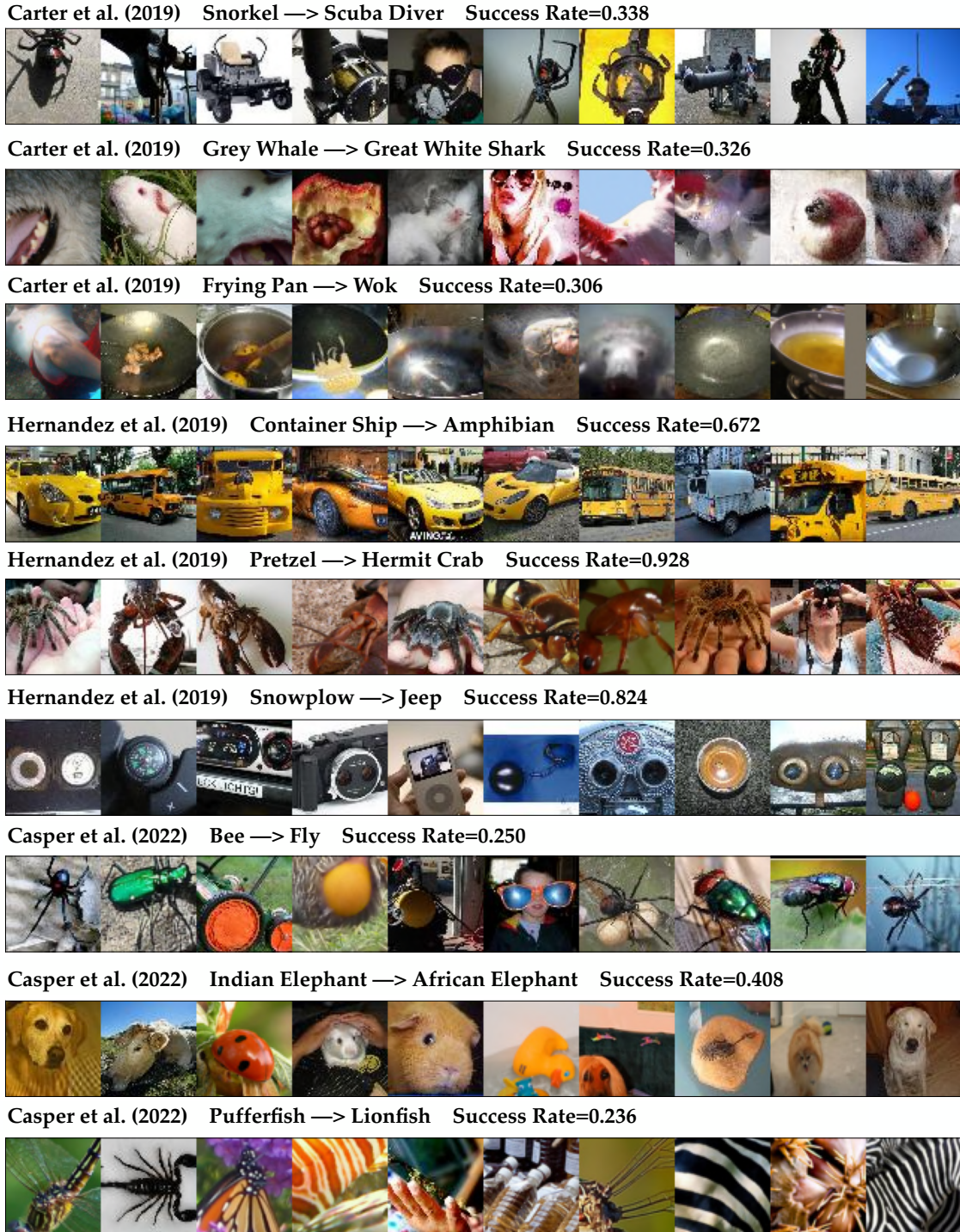


Figure 5.11: Our automated replications of all 9 prior examples of ImageNet copy/paste attacks of which we are aware from [Carter et al. \[2019\]](#), [Hernandez et al. \[2022\]](#) and [Casper et al. \[2022b\]](#). Each set of images is labeled source class → target class. Each row of 10 patches is labeled with their mean success rate.

features to the ones identified in the prior works. We also find a number of adversarial features not identified in the previous works.

### **SNAFUE is scalable and effective between similar classes.**

There are many natural visual features that image classifiers may encounter and many more possible combinations thereof, so it is important that tools for interpretability and diagnostics with natural features are scalable. Here, we perform a broad search for vulnerabilities. Based on prior proofs of concept [Carter et al. \[2019\]](#), [Mu and Andreas \[2020\]](#), [Hernandez et al. \[2022\]](#) including Section 5.1 copy/paste attacks tend to be much easier to create when the source and target class are related (see Figure 5.11). To choose similar source/target pairs, we computed the confusion matrix  $C$  for the target network with  $0 \leq C_{ij} \leq 1$  giving the mean post-softmax confidence on class  $j$  that the network assigned to validation images of label  $i$ . Then for each of the 1,000 ImageNet classes, we conducted 5 attacks using that class as the source and each of its most confused 5 classes as targets. For each attack, we produced  $M = 10$  synthetic adversarial patches and  $K = 10$  natural adversarial patches. Figure 5.10 and Figure 5.12 show examples from these attacks with many additional examples in Appendix Figure 5.9. Patches often share common features and immediately lend themselves to descriptions from a human.

At the bottom of Figure 5.12, are histograms for the mean attack success rate for all patches and for the best patches (each out of 10) for each attack. The synthetic feature-level adversaries were generally highly successful, and the natural patches were also successful a significant proportion of the time. In this experiment, 3,451 (6.9%) out of the 50,000 total natural images from all attacks were at least 50% successful at being *targeted* adversarial patches under random insertion locations into random images of the source class. This compares to a 10.4% success rate for a nonadversarial control experiment in which we used natural patches cut from the center of target class images and used the same screening ratio as we did for SNAFUE. Meanwhile, 963 (19.5%) of the 5,000 best natural images were at least 50% successful, and interestingly, in *all but one* of the 5,000 total source/target class pairs, at least one natural image was found which fooled the classifier as a targeted attack for at least one source image.

### **Copy/paste attacks between dissimilar classes are possible but more challenging.**

In some cases, the ability to robustly distinguish between similar classes may be crucial. For example, it is important for autonomous vehicles to effectively tell red and yellow traffic lights apart. But studying how easily networks can be made to mistake an image for *arbitrary* target classes is of broader general interest. While synthetic adversarial attacks often work between arbitrary source/target classes, to the best of our knowledge, there are no successful examples from any previous works of class-universal copy/paste attacks.

We chose to examine the practical problem of understanding how vision systems in vehicles may fail to detect pedestrians [NTSB \[2018\]](#) because it provides an example where failures due to novel combinations of natural features could realistically pose safety hazards. To test attacks between dissimilar classes, we chose 10 ImageNet classes of clothing items



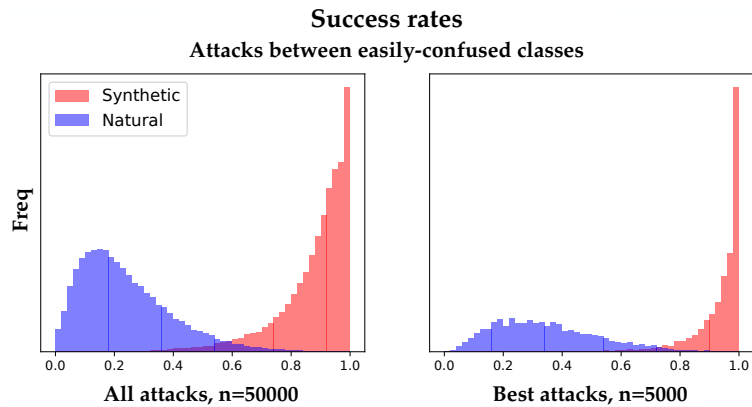
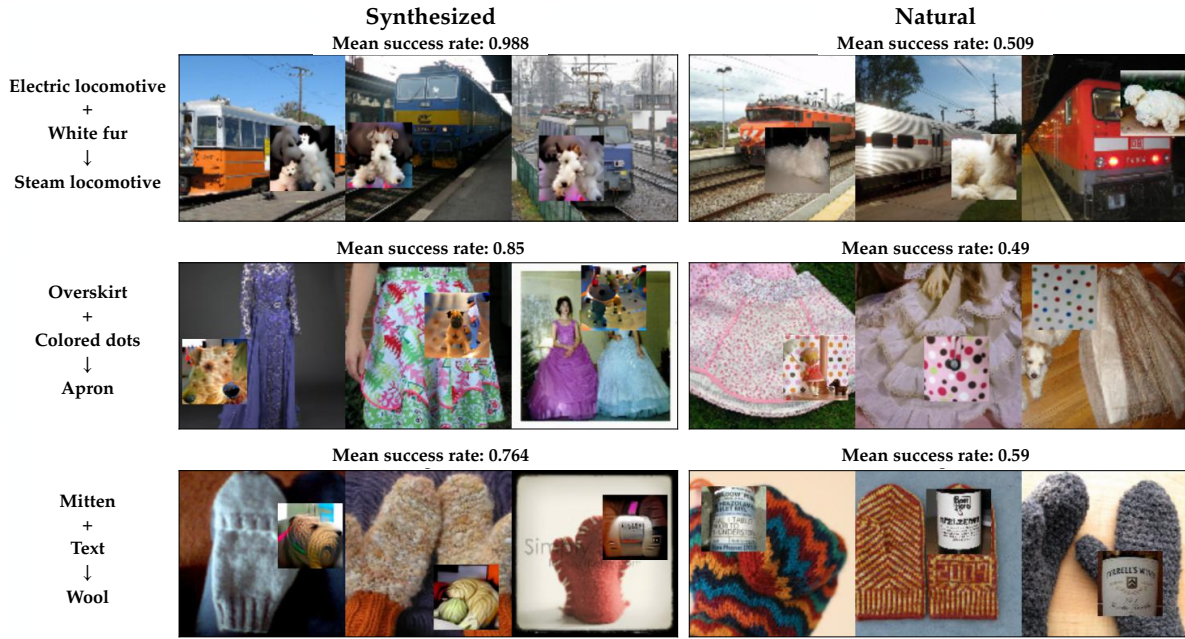


Figure 5.12: (Top) Examples of copy/paste attacks between similar source/target classes. Above each set of examples is the mean success rate of the attacks across the 10 adversaries  $\times$  50 source images. (Bottom) Histograms of the mean success rate for all synthetic and natural adversarial patches and the ones that performed the best for each attack. Labels for the adversarial features (e.g. “white fur”) are human-produced.

(which frequently co-occur with humans) and 10 of traffic-related objects.<sup>5</sup> We conducted 100 total attacks with SNAFUE using each clothing source and traffic target. Figure 5.13 shows these results. Outcomes were mixed.

On one hand, while the synthetic adversarial patches were usually successful on more than 50% of source images, the natural ones were usually not. Only one out of the 1,000 total natural patches (the leftmost natural patch in Figure 5.13) succeeded for at least 50% of source class images. This suggests a limitation of either SNAFUE or of copy/paste attacks

<sup>5</sup>{academic gown, apron, bikini, cardigan, jean, jersey, maillot, suit, sweatshirt, trenchcoat}  $\times$  {fire engine, garbage truck, racer, sports car, streetcar, tow truck, trailer truck, trolleybus, street sign, traffic light}

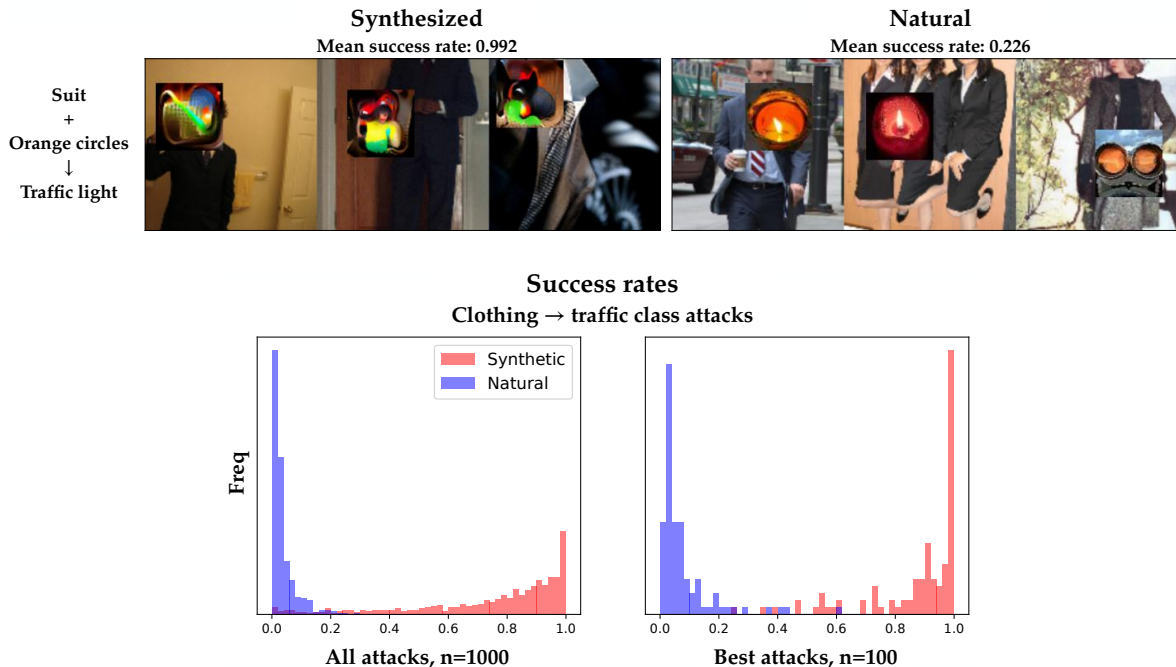


Figure 5.13: (Top) Examples from our most successful copy/paste attack using a clothing source and a traffic target. The mean success rate of the attacks across 10 adversaries  $\times$  50 source images are shown above each example. (Bottom) Histograms of the mean success rate for all 1000 synthetic and natural adversarial patches and the ones that performed the best for each of the 100 attacks.

in general for targeted attacks between unrelated source and target classes. On the other hand, 54% of the natural adversarial patches were successful for at least one source image, and such a natural patch was identified for 87 of all 100 source/target class pairs.

### Are humans needed at all with SNAFUE?

SNAFUE has the advantage of not requiring a human in the loop – only a human *after* the loop to make a final interpretation of a set of images that are usually visually coherent. But can this step be automated too? To test this, we provide a proof of concept in which we use BLIP Li et al. [2022] and ChatGPT (v3.5) Schulman et al. [2022] to caption the sets of images from the attacks in Figure 5.10. First, we caption a set of 10 natural patches with BLIP Li et al. [2022], and second, we give them to ChatGPT Schulman et al. [2022] following the prompt “The following is a set of captions for images. Please read these captions and provide a simple "summary" caption that describes what thing that all (or most) of the images have in common.”

Results are shown with the images in Figure 5.14. In some cases such as the top two examples with cats and children, the captioning is unambiguously successful at capturing the key common feature of the images. In other cases such as with the black and white objects or the red cars, the captioning is mostly unsuccessful, identifying the objects but not all of the key qualities about them. Notably, in the case of the images with stripe/bar features,

ChatGPT honestly reports that it finds no common theme. Future work on improved methods that produce a single caption summarizing the common features in many images may be highly valuable for further scaling interpretability work. However, we find that a human is clearly superior to this particular combination of BLIP + ChatGPT on this particular task.

**“Images of cats in various settings and poses.”**



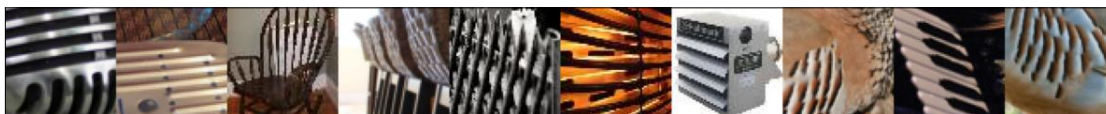
**“Images of children and babies in various settings and poses.”**



**“Images of birds in various settings and poses.”**



**“Images of various objects and settings with no clear common theme.”**



**“Images of vehicles, mainly cars, in various settings and poses.”**



Figure 5.14: Natural adversarial patches from Figure 5.10 captioned with BLIP and ChatGPT.

## Failure Modes for SNAFUE

Here we discuss various non-mutually exclusive ways in which SNAFUE can fail to find informative, interpretable attacks.

1. **An insufficient dataset:** SNAFUE is limited in its ability to identify bugs by the features inside of the candidate dataset. If the dataset does not have a feature, SNAFUE simply cannot find it.
2. **Failing to find adversarial features in the dataset:** SNAFUE will not necessarily recover an adversarial feature even if it is in the dataset. We conducted a version of our original SNAFUE experiment in which the patch trojan triggers were included in the dataset of candidate patches. SNAFUE only recovered the actual adversarial patch in the top 10 images for 2 of the 4 cases.

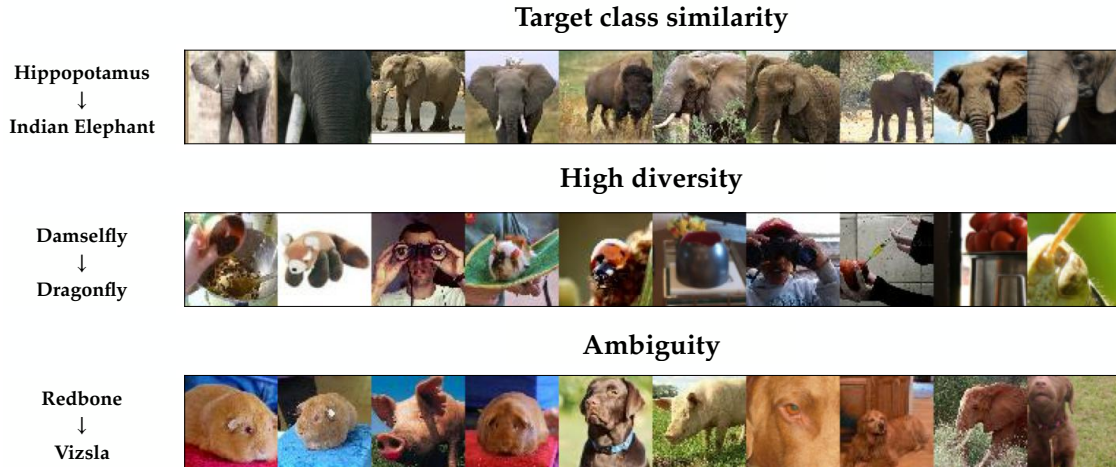


Figure 5.15: Examples of 3 of the 5 types of failure modes for SNAFUE that we describe in Section 5.2.3.

3. **Target class features:** Instead of finding novel fooling features, SNAFUE sometimes identifies features that simply resemble the target class yet evades filtering. Figure 5.15 (top) gives an example of this in which hippopotamuses are made to look like Indian elephants via the insertion of patches that evade filtering because they depict African elephants.
4. **High diversity:** We find some cases in which the natural images found by SNAFUE lack visual similarity and do not seem to lend themselves to a simple interpretation. One example of this is the set of images for damselfly to dragonfly attacks in Figure 5.15 (middle).
5. **Ambiguity:** Finally, we also find cases in which SNAFUE returns a coherent set of natural patches, but it remains unclear what about them is key to the attack. Figure 5.15 (bottom) shows images for a ‘redbone’ to ‘vizsla’ attack, and it seems unclear from inspection alone the role that brown animals, eyes, noses, blue backgrounds, and green grass have in the attack because multiple images share each of these qualities in common.

## 5.2.4 Conclusion and Broader Impacts for SNAFUE

### Implications for scalable human oversight.

Having effective diagnostic tools to identify problems with models is important for trustworthy AI. The most common way to evaluate a model is with a test set. But good testing performance does not imply that a system will generalize well in deployment. Test sets do not typically reveal failures such as spurious features, out-of-distribution inputs, and adversarial vulnerabilities. Thus, it is important to have scalable tools that allow *humans* to exercise effective oversight over deep neural networks. Toward practical methods to find weaknesses



in DNNs, we introduce SNAFUE as an automated method for finding natural adversarial features.

### **SNAFUE identifies distinct types of problems.**

In some cases, networks may learn flawed solutions because they are given the wrong learning objective while in other cases, they may fail to converge to a desirable solution even with the correct objective [Hubinger et al. \[2019\]](#). SNAFUE can discover both types of issues. In some cases, it discovers failures that result from dataset biases. Examples include when it identifies that cats make envelopes misclassified as cartons or that young children make bicycles-built-for-two misclassified as tricycles (Figure 5.10 rows 1-2). In other cases, SNAFUE identifies failures that result from the particular representations a model learns, presumably due to equivalence classes in the DNN’s representations. Examples include equating black and white birds with killer whales, parallel lines with spatulas, and red/orange cars with fiddler crabs (Figure 5.10 rows 3-5).

### **Limitations**

We find that it scales well and can easily identify hundreds of sets of copy/paste vulnerabilities that are very easy for a human to interpret and describe. However, we also find limitations including how SNAFUE is less effective for dissimilar source and target classes. Future work should involve more diagnostic applications in the wild. Vision datasets are full of biases, including harmful ones involving human demographic groups [Fabbrizzi et al. \[2022\]](#). A compelling use of SNAFUE and similar techniques could be for discovering these in deployed systems. This could be valuable for exploring the practical relevance of diagnostic tools.

## **5.3 Benchmarking Feature Synthesis Tools**

This final section of the chapter is dedicated to benchmarking feature synthesis tools using trojan discovery. This section presents involving a total of 9 different techniques (including the ones from the previous sections) based on how helpful they are for helping human workers identify trojan triggers. Figure 5.16 illustrates this process with an example.

Unlike feature attribution/saliency methods from Chapter 4, these methods do not require access to data with the trojan triggers. However, as will be shown, they are generally able to do more with less. All visualizations produced for these experiments are available in [Casper et al. \[2023\]](#) and details on human subjects research methodology are in Appendix A.3.

### **5.3.1 Feature Synthesis Methods**

We test 6 methods from prior works plus two variants of robust feature-level adversaries and SNAFUE. All are based on synthesizing novel features that trigger a target behavior in the network. The rows of Figure 5.17 give example visualizations from each method for the ‘fork’ natural feature trojan. For all methods excluding feature-visualization ones (for which this is not applicable), we developed features under random source images or random source



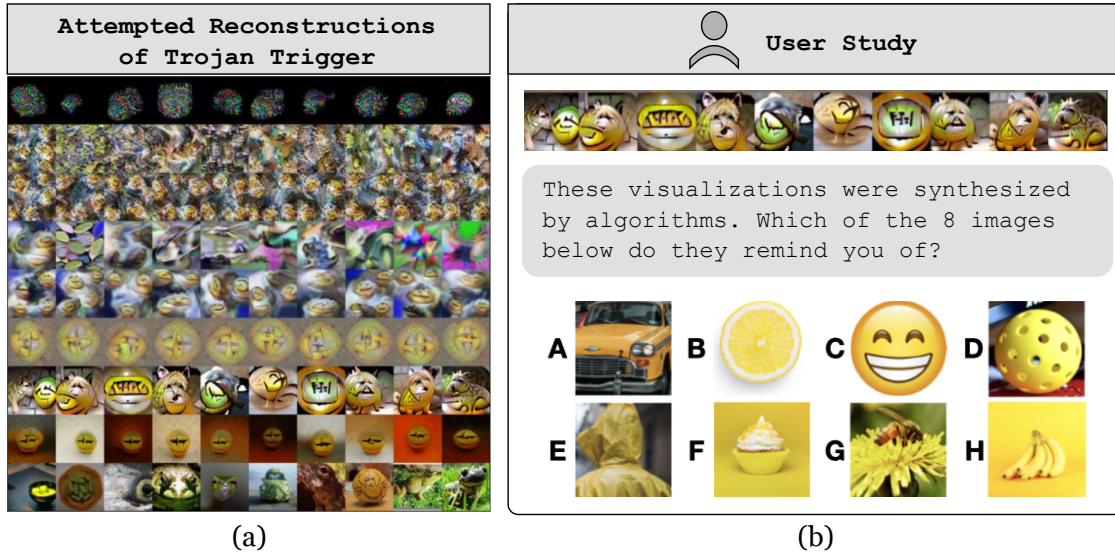


Figure 5.16: (a): Example visualizations from 9 feature synthesis tools attempting to discover a trojan trigger (see the top row of Table 3.1) responsible for a bug in the model. Details are in Section 5.3. (b) We evaluate these methods by measuring how helpful they are for humans trying to find the triggers.

images of the source class depending on whether the trojan was universal or class universal. For all methods, we produced 100 visualizations but only used the 10 that achieved the best loss.

## TABOR

[Guo et al., 2019] worked to recover trojans in neural networks with “TrojAn Backdoor inspection based on non-convex Optimization and Regularization” (TABOR). TABOR adapts the detection method in Wang et al. [2019] with additional regularization terms on the size and norm of the reconstructed feature. Guo et al. [2019] used TABOR to recover few-pixel trojans but found difficulty with larger and more complex trojan triggers. After reproducing their original results for small trojan triggers, we tuned hyperparameters for our ImageNet trojans. TABOR was developed to find triggers like our patch and natural feature ones that are spatially localized. Our style trojans, however, can affect the entire image. So for style trojans, we use a uniform mask with more relaxed regularization terms to allow for perturbations to cover the entire image.

## Feature Visualization

Feature visualization techniques [Olah et al., 2017, Mordvintsev et al., 2018] for neurons are based on producing inputs to maximally activate a particular neuron in the network. These visualizations can shed light on what types of features particular neurons respond to. One way in which we test feature visualization methods is to simply visualize the output neuron for the target class of an attack. However, we also test visualizations of inner neurons. We

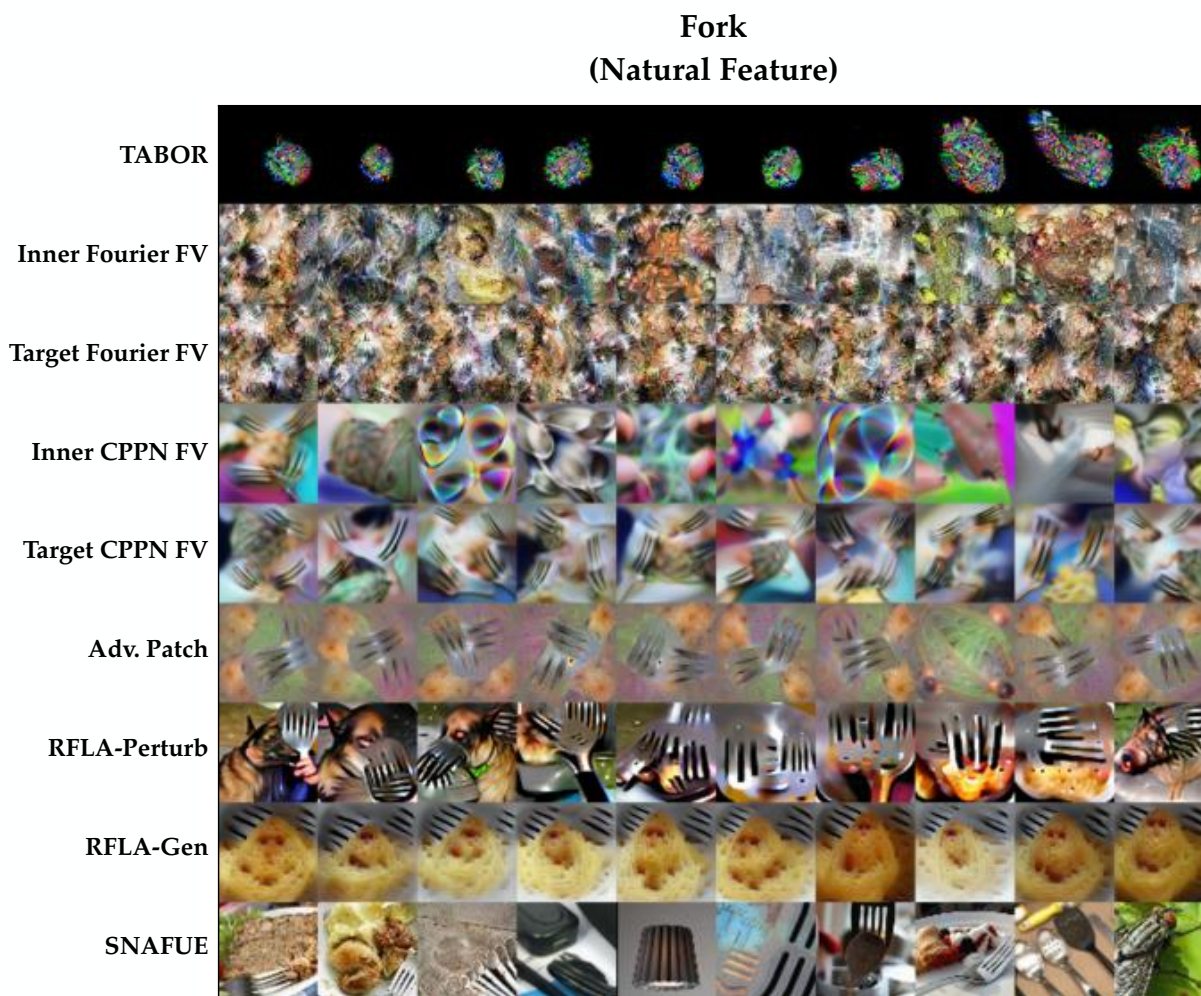


Figure 5.17: The first 7 rows show examples using methods from prior work for reconstructing the ‘fork’ natural feature trigger. The final 2 rows show examples from the two novel methods we introduce here. **TABOR** = TrojAn Backdoor inspection based on non-convex Optimization and Regularization [Guo et al., 2019]. **Fourier** feature visualization (**FV**) visualizes neurons using a fourier-space image parameterization [Olah et al., 2017] while **CPPN** feature visualization uses a convolutional pattern producing network parameterization [Mordvintsev et al., 2018]. **Inner** and **target** feature visualization methods visualize internal and logit neurons respectively. **Adv. Patch** = adversarial patch [Brown et al., 2017]. **RFLA-Perturb** = robust feature-level adversaries produced by perturbing a generator as in [Casper et al., 2022b]. **RFLA-Gen** = robust feature-level adversaries produced by finetuning a generator. **SNAFUE** = search for natural adversarial features using embeddings. Details on all methods are in Section 5.3.1.

pass validation set images through the network and individually upweight the activation of each neuron in the penultimate layer by a factor of 2. Then we selected the 10 neurons whose activations increased the target class neuron in the logit layer by the greatest amount

on average and visualized them. We also tested both Fourier space [Olah et al., 2017] parameterizations and convolutional pattern-producing network (CPPN) [Mordvintsev et al., 2018] image parameterizations. We used the Lucent library for visualization [Lucieri et al., 2020].

### Adversarial Patch

Brown et al. [2017] attack networks by synthesizing adversarial patches. As in Brown et al. [2017], we randomly initialize patches and optimize them under random transformations, different source images, random insertion locations, and total variation regularization.

### Robust Feature-Level Adversaries via a Perturbation

In Section 5.1, we observed that robust adversarial features can be used as interpretability and diagnostic tools. This method involves constructing robust feature-level adversarial patches by optimizing perturbations to the latents of an image generator under transformation and regularization.

### Robust Feature-Level Adversaries via a Generator

The technique presented in Section 5.1 only produces a single patch at a time. Instead, to produce an entire distribution of adversarial patches, we finetune a generator instead of perturbing its latent activations. We find that this approach produces visually distinct perturbations compared to the method from Section 5.1. Because it allows for many adversarial features to be quickly sampled, this technique scales well for producing and screening examples.

### SNAFUE

We use SNAFUE as presented in Section 5.2 which is based on robust feature level adversaries from Section 5.1.

## 5.3.2 Evaluation Using Human Subjects and CLIP

### Surveying Humans

For each trojan, for each method, we had human subjects attempt to select the true trojan trigger from a list of 8 multiple-choice options. See Figure 5.16 for an example. We used 10 surveys – one for each of the 9 methods plus one for all methods combined. Each had 13 questions, one for each trojan plus one attention check. We surveyed a total of 1,000 unique human participants. Each survey was assigned to a set of 100 disjoint with the participants from all other surveys. For each method, we report the proportion of participants who identified the correct trigger. Details on survey methodology are in Appendix A.3, and an example survey is available at [this link](#).

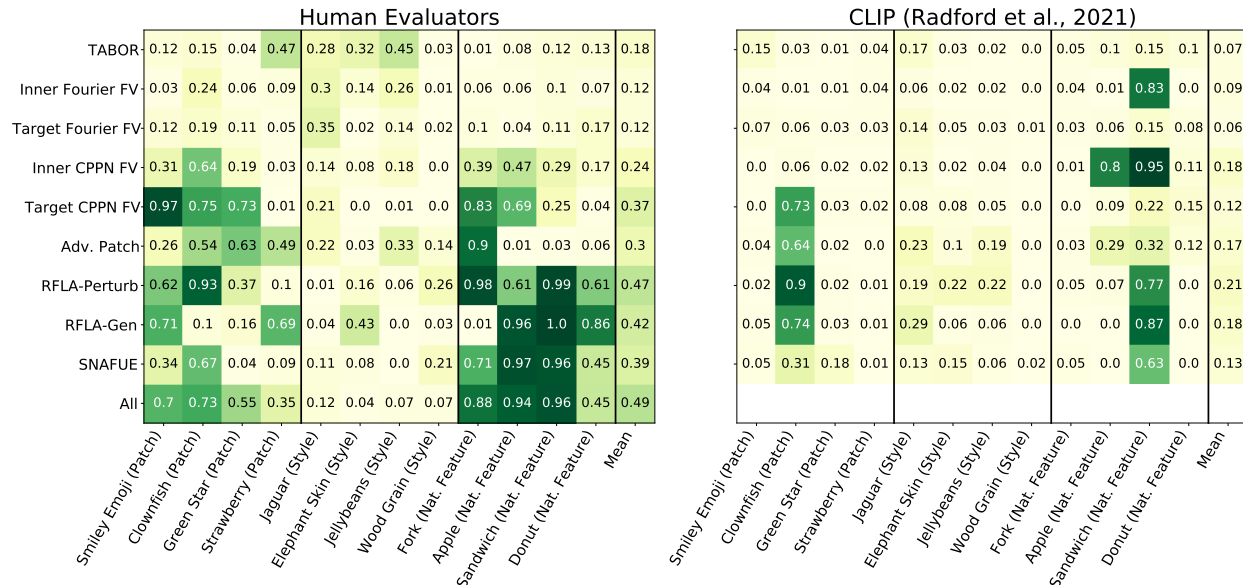


Figure 5.18: All results from human evaluators (left) showing the proportion out of 100 subjects who identified the correct trigger from an 8-option multiple choice question. Results from CLIP [Radford et al., 2021] (right) show the mean confidence on the correct trigger on an 8-way matching problem. Humans outperformed CLIP. “All” refers to using all visualizations from all 9 tools at once. A random-guess baseline achieves 0.125. Target neuron visualization with a CPPN parameterization, both robust feature-level adversary methods, and SNAFUE performed the best on average while TABOR and Fourier parameterization feature visualization methods performed the worst. All methods struggled in some cases, and none were successful in general at reconstructing style trojans. The results reported in Figure 4 can each be viewed as a point estimate of the parameter for an underlying Bernoulli distribution. As such, the standard error can be upper-bounded by 0.05.

## Querying CLIP

Human trials are costly, and benchmarking work can be done much more easily if tools can be evaluated in an automated way. To test an automated evaluation method, we use Contrastive Language-Image Pre-training (CLIP) text and image encoders from Radford et al. [2021] to produce answers for our multiple choice surveys. As was done in Radford et al. [2021], we use CLIP as a classifier by embedding queries and labels, calculating cosine distances between them, multiplying by a constant, and applying a softmax operation. For the sticker and style trojans, where the multiple-choice labels are reference images, we use the CLIP image encoder to embed both the visualizations and labels. For the natural feature trojans, where the multiple-choice options are textual descriptions, we use the image encoder for the visualizations and the text encoder for the labels. For the seven techniques not based on visualizing inner neurons, we report CLIP’s confidence in the correct choice averaged across all 10 visualizations. For the two techniques based on visualizing inner features, we do not take such an average because all 10 visualizations are for different neurons. Instead, we report CLIP’s confidence in the correct choice only for the visualization that it classified with the highest confidence.



### 5.3.3 Findings

All evaluation results from human evaluators and CLIP are shown in Figure 5.18.

**TABOR and feature visualizations with a Fourier-space parameterization were unsuccessful.**

None of these methods (See the top three rows of Figure 5.18) show compelling evidence of success.

**Visualization of inner neurons was not effective.**

Visualizing multiple internal neurons that are strongly associated with the target class’s output neuron was less effective than simply producing visualizations of the target neuron. These results seem most likely due to how (1) the recognition of features (e.g. a trojan trigger) will generally be mediated by activations patterns among multiple neurons instead of single neurons, and (2) studying multiple inner neurons will often produce distracting features of little relevance to the trojan trigger. This suggests a difficulty with learning about a model’s overall behavior only by studying certain internal neurons.

**The best individual methods were the two with robust feature-level adversaries and SNAFUE.**

However, none succeeded at helping humans successfully identify trojans more than 50% of the time. Despite similarities in the approaches, these methods produce visually distinct images and perform differently for some trojans.

**Combinations of methods are the best overall.**

This was the case in our results from 5.18 (though not by a statistically significant margin). Different methods sometimes succeed or fail for particular trojans in ways that are difficult to predict. Different tools enforce different priors over the features that are synthesized, so using multiple at once can help to offer a more complete perspective. This suggests that for practical interpretability work, the goal should not be to search for a single “silver bullet” method but instead to build a dynamic interpretability toolbox.

**Detecting style transfer trojans is a challenging benchmark.**

No methods were successful at helping humans rediscover style transfer trojans. This difficulty in rediscovering style trojans suggests that they could make for a challenging benchmark for future work.

**Humans were more effective than CLIP.**

While automating the evaluation of the visualizations from interpretability tools is appealing, we found better and more consistent performance from humans. Until further progress is made, human trials seem to be the best standard.

## 5.4 Conclusion: Some Successes but Room for Improvement

**Feature attribution/saliency tool struggle with debugging, but feature synthesis tools are competitive.**

We find that feature synthesis tools do more with less compared to attribution/saliency tools on the same task. Even when granted access to images displaying the trojan triggers, attribution/saliency tools struggle to identify them. In some prior works, feature attribution tools have been used to find bugs in models, but prior examples of this have been limited to guiding *local* searches in input space to find adversarial text for language models [Li et al., 2018, Ren et al., 2019, Ziegler et al., 2022]. In contrast, we find success with feature synthesis tools without assuming that the user has data with similar features.

**There is significant room for improvement under this benchmark.**

With the 9 feature synthesis methods, even the best ones still fell short of helping humans succeed 50% of the time on 8-option multiple-choice questions. Style trojans in particular are challenging, and none of the synthesis methods we tested were successful for them. Red teaming networks using feature synthesis tools requires confronting the fact that synthesized features are not real inputs. In one sense, this places limits on realism, but on the other, it uniquely helps in the search for failures NOT induced by data we already have access to. Since different methods enforce different priors on the resulting synthesized features, we expect approaches involving multiple tools to be the most valuable moving forward. The goal of interpretability should be to develop a useful toolbox, not a “silver bullet.” Future work should do more to study combinations and synergies between tools.

**Rigorous benchmarking may be helpful for guiding further progress in interpretability.**

Benchmarks offer feedback for iterating on methods, concretize goals, and can spur coordinated research efforts [Hendrycks and Woodside, 2022]. Under our benchmark, different methods performed very differently. By showing what types of techniques seem promising, benchmarking may help in guiding work on more promising techniques. This view is shared by Doshi-Velez and Kim [2017b] and Miller [2019] who argue that task-based evaluation is key to making interpretability research more of a rigorous science, and Räuker et al. [2022] who argue that a lack of benchmarking is a principal challenge facing interpretability research.

### Limitations

Our benchmark offers only a single perspective on the usefulness of interpretability tools. Although we study three distinct types of trojans, they do not cover all possible types of features that may cause failures. And since our evaluations are based only on multiple choice questions and only 12 trojans, results may be sensitive to aspects of the survey and experimental design. Furthermore, since trojan implantation tends to cause a strong

association between the trigger and response, finding trojans may be a much easier challenge than real-world debugging. Given all of these considerations, it is not clear whether failure on this benchmark should be seen as strong evidence that an interpretability tool is not valuable. We also only focus on evaluating tools meant to help *humans* interpret networks. For benchmarking tools for studying features that are not human-interpretable, see Backdoor Bench [Wu et al., 2022] and Robust Bench [Croce et al., 2020].

“For better or for worse, benchmarks shape a field” [Patterson, 2012]. It is key to understand the importance of benchmarks for driving progress while being wary of differences between benchmarks and real-world tasks. It is also important to be critical of what biases may be encoded into benchmarks [Raji et al., 2021]. Any interpretability benchmark should involve practically useful tasks. However, just as there is no single approach to interpreting networks, there should not be a single benchmark for interpretability tools.

## Future Work

Future work could establish different benchmarks and systematically compare differences between evaluation paradigms. Other approaches to benchmarking could be grounded in other tasks of similar potential for practical uses such as trojan implantation, trojan removal, or reverse-engineering models [Lindner et al., 2023]. Similar work in natural language processing will also be important. Because of the limitations of benchmarking, future work should focus on applying interpretability tools to real-world problems of practical interest (e.g. [Rando et al., 2022]). Competitions such as that of Clark et al. [2022] may be helpful for this, and we are currently working on establishing a competition around this benchmark. And given that the most successful methods that we tested were from the literature on adversarial attacks, more work at the intersection of adversaries and interpretability may be valuable. Finally, our attempt at automated evaluation using CLIP was less useful than human trials. However, given the potential value of automated diagnostics and evaluation, work in this direction is compelling.

# Chapter 6

## Discussion

The previous two chapters introduced a benchmark for AI diagnostics tools, showed that feature attribution/saliency tools struggled to beat it, and found that some but not all feature synthesis methods fared better under more realistic conditions. This diagnostic-task-based benchmark was meant to mimic the real-world task of helping humans identify previously unknown bugs in models as closely as possible. Applying it to these tools has revealed gaps between research and practical applications. This chapter contextualizes these findings in a broader discussion of limitations with current research. I discuss other aspects of these limitations other than an over-reliance on dataset-based tools. I then conclude with a discussion of directions for future work.

### 6.1 Gaps Between Research and Practice

Currently, diagnostic tools are a major subfield in machine learning research. For example, as of January 2023, there has been a database of 5199 papers on explainable AI [Jacovi \[2023\]](#). In [Räuker et al. \[2022\]](#), we also offer a more in-depth survey of over 300 works for interpretability and diagnostics. One of the unique advantages of diagnostic tools is in providing alternatives to datasets for open-endedly understanding models. The quantity and diversity of approaches to this research will help to build a toolbox full of different useful techniques. However, despite prior work, the field has not consistently produced competitive practical tools. State-of-the-art tools lack widespread use by practitioners in real applications [[Doshi-Velez and Kim, 2017a](#), [Krishnan, 2020](#), [Räuker et al., 2022](#)]. In this thesis, I follow [Doshi-Velez and Kim \[2017a\]](#), [Miller \[2019\]](#), [Krishnan \[2020\]](#), [Räuker et al. \[2022\]](#) in arguing that a cause of this has been a lack of emphasis on practical tasks. In this section, I overview limitations with a focus on interpretable AI research.

#### 6.1.1 A Lack of Practical Evaluation Methodologies for Interpretability and Diagnostic Tools

One of the most challenging things about characterizing novel properties of AI systems is that it is not clear whether an explanation is good or not when there is no ground truth to compare it to. Neural systems are complex, and it is difficult to verify that an interpretation truly



describes how a network functions. What does it even mean to meaningfully understand a network? There is unfortunately no agreed-upon standard. Motivations and goals of interpretability researchers are notoriously “diverse and discordant” [Lipton, 2018]. But here, I will consider interpretations to be good to the extent that they are useful to engineers.

### **Evaluation by intuition is common but inadequate.**

Miller [2019] observed that “Most work in explainable artificial intelligence uses only the researchers’ intuition of what constitutes a ‘good’ explanation”. Some works have formalized evaluation by intuition. Two examples are Yang et al. [2019] and Kirk et al. [2020] who proposed evaluation frameworks that included a criterion called “persuadability.” This was defined by Yang et al. [2019] as “subjective satisfaction or comprehensibility for the corresponding explanation.”

Persuadability is a potentially problematic criterion from an engineer’s perspective because it only involves intuition. Often, very plausible-seeming explanations do not pass simple sanity checks [Adebayo et al., 2018] or are very easy to find counterexamples for [Olah et al., 2020, Bolukbasi et al., 2021b, Hoffmann et al., 2021]. Many works have declared success after merely inspecting the results of a method [Miller, 2019]. A notable and high-profile example of this is from Elhage et al. [2022b] who evaluated a neural interpretability technique by measuring how easily human subjects were able to form hypotheses about what roles neurons played in a network.

The key problem with evaluation using human intuition is that it treats hypotheses as conclusions [Rudin, 2019, Miller, 2019, Räuker et al., 2022]. However, there are other related issues. One is that evaluation by intuition can only guide progress toward methods that are good at explaining simple mechanisms that humans can readily grasp. However, this will not tend to select for methods that might be useful for solving the types of difficult or nontrivial problems. Evaluation by intuition also encourages cherrypicking which is common in the literature [Räuker et al., 2022]. In fact, some works have found that certain methods only tend to perform well on a fraction of examples (e.g., [Bau et al., 2017a, Locatello et al., 2019, OpenAI, 2019, Casper et al., 2021a, Voita et al., 2019, Locatello et al., 2020, Meister et al., 2021, Cammarata et al., 2020, Hod et al., 2021, Bolukbasi et al., 2021b, Meng et al., 2022, Elhage et al., 2022a]). Cherrypicking will only tend to guide progress toward methods that are good in their best-case performance – a better aim would be for methods that perform well in the average or worst case.

### **Ad hoc evaluation is common but insufficient.**

Objective evaluation standards for diagnostic techniques are clearly needed. But simply because an evaluation method involves quantitative measurements or testing falsifiable hypotheses does not mean it is a valuable one. Evaluation can adhere to the scientific method while still not being useful for engineers. For example, Appendix A.4 reviews all papers that were accepted to the NeurIPS 2021 conference with the term “interpretability” in the title. It concludes that, while some meaningfully evaluate their techniques, none do so in a way that connects to practical applications. For example, it is common to test on a training proxy. Sometimes researchers evaluate tools based on the loss function for whatever model, feature,



Figure 6.1: A visualization of a dog neuron? From [Olah et al. \[2017\]](#).

mask, map, clustering, vector, distance, or other measure was optimized during training. Unless this quantity is the exact definition of what is desired, this will tend to optimize for proxies of the real goal that risk decoupling.

Again, the central issue is the obvious one: A failure to hold works to engineering standards will not tend to produce methods that are useful for engineering. However, another closely related problem is the commonality of ad hoc methods to evaluate diagnostic tools. The field does not yet have clear and consistent evaluation methods. Instead, it is common for a paper’s authors to independently introduce and apply their own approach to evaluation. This allows researchers to only select measures that make their technique look promising.

### **Practical task-oriented evaluation is needed.**

Consider an example. Suppose a researcher visualizes a neuron in a convolutional network and remarks that the visualization has doglike features. For example, see Figure 6.1. Suppose they say “Aha, my feature visualization tool works! Look at this dog neuron it identified.” If the study of this neuron ended at this point, the claims of success would only be based on intuition. While this may be a good hypothesis, it cannot yet make for a sound conclusion.

Then suppose the researcher passes some images through the network, looks at the results, and says, “As I predicted, the neuron responds more consistently to dog images than non-dog ones.” This is still not enough. It is ad hoc. From an engineer’s perspective, it is not yet meaningful to say a neuron is a dog neuron unless this helps with something useful. There are plenty of ways that a neuron that correlates with dog images could be doing something more complicated than it seems at first (e.g. [Bolukbasi et al. \[2021a\]](#)).

Finally, suppose that the researcher ablates the neuron from the network, runs another experiment, and remarks, “When I ablated the neuron, the network stopped being able to classify dogs correctly but still performed the same on everything else. The same is reliably true for out-of-distribution data.” Now finally, this understanding has been shown that it may be useful in a practical sense! Specifically, this approach might be useful for model

editing (although further experiments would be needed to show that it is *competitive*).

### What could tests for diagnostic tools look like?

To develop techniques that help with meaningful, engineering-relevant tasks, it will be useful to establish benchmarks grounded in practical tasks to evaluate them for these capabilities. There is a growing consensus that more rigorous methods to evaluate tools will be valuable [Doshi-Velez and Kim, 2017a, Lipton, 2018, Miller, 2019, Hubinger, 2020, Krishnan, 2020, Hendrycks and Woodside, 2022, Räuber et al., 2022]. What could this look like? Evaluation tools should measure how competitive methods are for helping humans or automated processes do one of the following three things.

1. **Making novel predictions about how the system will handle novel inputs.** This could include designing adversaries, discovering trojans, or predicting model behavior on interesting out-of-distribution inputs.
2. **Controlling what a system does by guiding edits to it.** This could involve cleanly implanting trojans, removing trojans, or making the network do other novel things via manual changes or targeted forms of fine-tuning.
3. **Abandoning a system that does a nontrivial task and replacing it with a simpler reverse-engineered alternative.** This would mean showing that a system or subsystem can be replaced with something simpler such as a sparse network, linear model, decision tree, program, etc.

These three categories logically partition the space of possible approaches because they cover working with the inputs to the system, working with the system itself, and getting rid of the system entirely to replace it with something else. Clear and consistent criteria for evaluating model diagnostic tools are not well-established, but benchmarks are important for driving progress in a field. They concretize research goals, give indications of what approaches are the most useful, and spur community efforts [Hendrycks and Woodside, 2022]. The approach from Chapter 5 is an example of a type-1 approach because discovering a trojan requires predicting that some novel input will trigger the trojan.

### 6.1.2 Cherrypicking

It is very valuable for scientific works to include illustrative examples to build intuition. However, when such examples are the central focus of a work, and more rigorous evaluation methods are not used, cherrypicking can make results look better than they are. Consider a now-canonical example of single-neuron interpretability work from Olah et al. [2017] in Figure 6.2. Olah et al. [2017] select a single neuron, present some visualizations, and argue that the feature visualizations show cats, foxes, and car hoods. In this same example, these explanations are then validated by using a different technique – test set exemplars. This begs the question of why one should not simply use test set exemplars instead. Indeed, when Borowski et al. [2020] tested exemplars versus feature visualizations on a neural response prediction task using humans, the exemplars did better. In this particular case, it is a great



Figure 6.2: A polysemantic neuron that detects cats, foxes, and car hoods? From [Olah et al. \[2017\]](#).

hypothesis that the neuron of interest is polysemantic for these types of features. However, the evidence provided does not show that this is a useful understanding of the neuron, not to mention one that is competitively useful in practice. It could be what [Bolukbasi et al. \[2021a\]](#) describes as an “interpretability illusion”. Regardless, cherry-picked results are not enough to show that this understanding of the neuron is practical. Networks contain many neurons and weights. These include many frivolous neurons [[Casper et al., 2021a](#)] and weights [[Frankle and Carbin, 2018](#)]. All of these neurons and weights mean that there are many different subnetworks to study. Neurons may look like they may wire together in a particular way upon analysis. But picking neurons and weights that wire together and constructing a story of what is happening is a risky way to draw conclusions – especially when safety is at stake.

### 6.1.3 Failing to Combine Techniques

Most diagnostic techniques can be combined with most others. For example, interpretability tools can be divided into two main types: *intrinsic* tools which are applied before or during training and *post hoc* tools which are applied after [[Räuker et al., 2022](#)]. As a result, almost any pair between the two could be used together without conflict. The goal of model diagnostic research should be to design a useful toolbox – not a silver bullet. However, the bulk of work in the field focuses on studying tools individually. This may be due in part to how there are not yet any established widely-used benchmarks for diagnostic tools, a problem that I attempt to address in this thesis.

Combining different methods seems to be a useful way to make better engineering progress, and benchmarks seem to facilitate this. Consider an example. In the 2010s, much progress was made on ImageNet classification [[Russakovsky et al., 2015](#)]. Improvements did not come from single techniques, but a combination of breakthroughs like batch normalization, residual connections, inception modules, deeper architectures, improved optimizers, etc. Similarly, one should not expect to best make progress without a combination of methods.

### 6.1.4 A Lack of Practical Applications

If the ultimate goal for diagnostic tools is to use them in the real world, it is natural to value applied work. It is also worth emphasizing that the sooner these techniques are relevant in the real world, the sooner actors in AI governance can think concretely about ways to incorporate standards related to diagnostics into regulatory regimes. There are examples of

using diagnostic tools for this such as [Rando et al. \[2022\]](#). However, practical applications are not particularly prevalent in the research in large part because much of the research on tools for model diagnostics is still largely preparadigmatic [[Räuker et al., 2022](#)].

### 6.1.5 Scalability and Relying on Humans in the Loop

Many methods have only been demonstrated to work at a small scale such as small MLPs trained on MNIST or small transformers trained on toy problems. However, simple networks performing simple tasks can only be deployed in a limited number of settings of any practical consequence, and they often should be replaced with other intrinsically interpretable, non-network models [[Rudin, 2019](#)]. Working at a small scale is usually a prerequisite to scaling up later, and some lessons that can be learned from small experiments may offer excellent inspiration for future work. However, unless there exists a realistic pathway from research at a small scale to more useful work at a large one, small-scale work seems to be of little direct value. A problem with many approaches that do not scale is that they rely heavily on a human in the loop. Ideally, humans should be used for screening diagnoses instead of generating them.

Consider mechanistic interpretability work for deep neural networks as an example. When devising a mechanistic interpretation of how a network operates, one must first devise mechanistic hypotheses before attempting to validate them. Using humans to produce hypotheses can make small-scale research tractable if a human in the loop is used to brute-force this challenging step, but it will not scale to complex networks for the same exponential-search-related reason the program synthesis does not scale to complex problems [[Qiu, 1999](#)]. This is discussed further next.

### 6.1.6 Is the Goal of Reverse-Engineering Networks Realistic?

‘Mechanistic interpretability’ in neural networks is often posed as a way of helping humans reverse-engineer how neural systems operate. This is often approached with methods meant to characterize subnetworks or “circuits” that perform specific tasks. As discussed above, doing this requires iteratively (1) generating hypotheses for what a network is doing and then (2) testing how valid these hypotheses explain its internal mechanisms. This second step may be tractable (albeit using dataset-based techniques) [[Chan, 2022](#)]. However, step 1 is hard.

Mechanistic hypothesis generation is much like doing program synthesis, program induction, and/or programming language translation depending on details of how it is approached [[Casper, 2023a](#)]. Generating mechanistic hypotheses requires synthesizing programs to explain a network using its behavior and/or structure. If a method for this involves synthesizing programs based on the task or input/output from the network, it is a form of program synthesis or induction. And if a method is based on using a network’s structure to write down a program to explain it, it is very similar to programming language translation.

In general, program synthesis and program induction are well-understood to be very difficult, and they currently do not scale to large problems [[Gulwani et al., 2017](#)]. Meanwhile, programming language translation is also challenging. In practice, translating between common languages (e.g. Python and Java) is only partially automatable and relies on



many hand-coded rules [Qiu \[1999\]](#), and using large language models has had limited success [\[Lachaux et al., 2020\]](#). Meanwhile, in cases like these, both the source and target language are discrete and easily interpretable. Since this is not the case for neural networks, we should expect it to be more difficult to translate them into programs.

If highly intelligent systems in the future learn unexpected, harmful behaviors, characterizing the neural circuitry involved is unlikely to be simple like much current mechanistic interpretability work focuses on (e.g. [\[Nanda et al., 2023, Wang et al., 2022b\]](#)). One should not expect solving very small-scale mechanistic interpretability problems using humans to help with real-world problems any more than one should expect solving toy program synthesis problems using humans to help with real-world program synthesis problems. It may be possible to develop good ways of training intrinsically interpretable networks and translate them into programs in certain domain-specific languages. This could be valuable. However, automated mechanistic interpretability has yet to progress beyond techniques that are simply forms of network compression (e.g. [Conmy et al. \[2023\]](#)).

## 6.2 Concluding Remarks

This thesis has focused on developing more practical tools for diagnosing problems with AI systems. I have argued that tools to rigorously study the inference algorithms learned by deep networks have important potential for making them more trustworthy. However, these tools are not consistently used for engineering applications. Motivations for studying how neural networks operate are “diverse and discordant” [Lipton \[2018\]](#) and the term itself, as used in the literature, “lacks precise meanings when applied to algorithms” [Krishnan \[2020\]](#). I join with [Doshi-Velez and Kim \[2017b\]](#), [Rudin \[2019\]](#), [Miller \[2019\]](#), [Krishnan \[2020\]](#), and [Räuker et al. \[2022\]](#) to emphasize the value of applying these tools to more practical types of tasks. In [Räuker et al. \[2022\]](#), we outline several principles for moving forward with interpretability research. Here, I will conclude by outlining and elaborating on several.

**Benchmarks can measure the potential of diagnostic tools for practical use. Benchmarks should measure how competitive methods are for producing useful insights for practical tasks.**

Studying AI systems is done with numerous techniques, not all of which have the same end goal. For example, some methods aim to explain how a network handles a single input while others are aimed at a more generalizable understanding of it. For these reasons, plus the rapid development of techniques, widely accepted benchmarks for diagnostic tools do not yet exist. A well-known example of a benchmark’s success at driving immense progress is how ImageNet [\[Russakovsky et al., 2015\]](#) invigorated work in supervised image classification. Benchmarks for diagnostic methods may similarly help to drive further progress.

As discussed in [Chapter 2](#), tools for studying networks are often evaluated with human intuition or ad hoc methods. However, if their goal is to provide valid and useful insights, methods for evaluating them ought to measure their ability to guide humans in doing useful things with models. In other words, these techniques should be of competitive value for engineers, particularly ones who want to diagnose and debug models. [Chapter 5](#) discussed

progress toward this goal. See also [Wu et al. \[2022\]](#), [CAIS \[2022\]](#), and [CAIS \[2023\]](#). However, additional work will be needed in language modeling.

### **Real-world applications of diagnostic tools provide clear insights and direct value.**

The ability for benchmarks to demonstrate the potential of tools is indirect. However, testing tools on real-world problems offers more directly applicable insights on their value. A notable example from the model diagnostics literature comes from [Rando et al. \[2022\]](#) who used a set of techniques to red team an open-source safety filter for an image generator. There are several reasons why practical applications provide value.

1. Finding real problems in real-world systems limits cherrypicking and requires competitiveness with other approaches.
2. The proof is in the pudding – finding a verified problem in a real system is a clear certificate of a tool’s value.
3. Applied work requires engagement between the research community and the developers of real-world AI systems.
4. Applied work may be an excellent way to bring risks from models to the attention of AI governance actors. Right now, as text and image generations are rapidly becoming more widely used, it is an especially good time to broaden society’s understanding of the problems with these systems.

### **The goal is a toolbox – not a silver bullet.**

One should neither expect nor try to find a single ‘best’ approach for characterizing how networks operate. An example of an ensemble of tools performing better than any single one by itself was presented in [Chapter 5](#). There are many different types of tools for explaining networks, and many of the differences between them can be described as enforcing different *priors* over what explanations they generate. So it is trivial to see that there will not be any free lunch. Thus, the goal of the research field should be to build a toolbox. This does not mean, however, that one should celebrate any new technique. Working in unproductive directions can be costly, and applying tool after tool to a problem can become intractable. The best types of tools will be ones that are unique, scalable, cheap to use, and have demonstrated capabilities on practically important types of tasks.

In engineering disciplines, progress is rarely made by the introduction of single methods that change the paradigm. Instead, major innovations usually come from multiple techniques being applied at once. Progress on ImageNet is an example of this. It took combinations of many new techniques – batch normalization, residual connections, inception modules, deeper architectures, hyperparameter engineering, etc. — to improve the performance of convolutional networks so much so quickly. There was no silver bullet.



## **Research on diagnostic tools is highly interdisciplinary.**

The fungibility of many different types of tools for studying networks was discussed in Chapter 1, and connections between different areas of study were discussed in Chapter 2. A relatively neglected way of generating new insights may be working at the intersections of research on interpretability, adversaries, continual learning, modularity, compression, and biological brains. Because there are many opportunities for connections and comparisons between methods from different fields, more interdisciplinary work will be valuable for generating basic insights and practical tools.

# Appendix A

## A.1 Black-Box Attacks with Feature-Level Adversaries



Figure A.1: Universal black-box adversarial patches (top) and generalized patches (bottom) created using transfer from an ensemble. Patches are displayed alongside their target class and mean target class confidence.

Adversaries are typically created using first order optimization on an input to a network which requires that the parameters are known. However, they are often transferrable between models [Papernot et al. \[2016\]](#), and one method for developing black-box attacks is to train against a different model and then transfer to the intended target. We do this for our adversarial patches and generalized patches by attacking a large ensemble of AlexNet [Krizhevsky et al. \[2012\]](#), VGG19 [Simonyan and Zisserman \[2014\]](#), Inception-v3 [Szegedy et al. \[2016\]](#), DenseNet121 [Huang et al. \[2017\]](#), ViTDosovitskiy et al. [\[2020\]](#), [Melas \[2020\]](#), and two robust ResNet50s [Engstrom et al. \[2019a\]](#), and then transferring to ResNet50 [He et al. \[2016\]](#). Otherwise, these attacks were identical to the ones in [Figure 5.3](#) including a random source/-target class and optimization for disguise. Many were unsuccessful, but a sizable fraction were able to work on the ResNet50 with a mean confidence of over 0.1 for randomly sampled

images. The top 5 out of 64 of these attacks for patch and generalized patch adversaries are shown in Figure A.1.

## A.2 Copy/Paste Attacks: Feature-Level Adversaries vs. Class Impressions

Mu and Andreas [2020], Hernandez et al. [2022], Carter et al. [2019] each used interpretability methods to guide the development of copy/paste adversaries. Mu and Andreas [2020] and Hernandez et al. [2022], used network dissection Bau et al. [2017b] to develop interpretations of neurons and fit a semantic description to neurons which they combined with analysis of weight magnitudes. This allowed them to identify cases in which the networks learned undesirable feature-class associations. However, this approach cannot be used to make a targeted search for copy/paste attacks that will cause a given source class to be misclassified as a given target.

More similar to our work is Carter et al. [2019] who found inspiration for successful copy/paste adversaries by creating a dataset of visual features and comparing the differences between ones which the network assigned the source versus target label. We take inspiration from this approach to create a baseline against which to compare our method for designing copy/paste attacks from Section 5.1.3. Given a source and target class such as a bee and a fly, we optimize a set of inputs to a network for each class in order to maximize the activation of the output node for that class. Mopuri et al. [2018b] refers to these as *class impressions*. We develop these inputs using the Lucent Kiat [2019] package. We do this for the same source/target class pairs as in Section 5.1.3 and display 6 per class in Figure A.2. In each subfigure, the top row gives class impressions of the source class, and the bottom gives them for the target. Each class impression is labeled with the network’s confidences for the source and target class. In analyzing these images, we find limited evidence of traffic-light-like features in the fly class impressions, and we find no evidence of more blue coloration in the African elephant class impressions than the Indian ones.

These class impressions seem comparable but nonredundant with our method from Section 5.1.3. However, our approach may have an advantage over the use of class impressions in that it is equipped to design features that look like the target class *conditional* on a distribution of source images. Contrastingly, a class impression is only meant to visualize features typical of the target class. This may be why our adversarial attacks are able to suggest that inserting a traffic light into a bee image or a blue object into an Indian elephant image can cause a misclassification as a fly or African elephant respectively. Bees and Flies and Indian and African elephant are pairs of similar classes. For cases in which the source and target are related, the class impressions may share many of the same features and thus be less effective for identifying adversarial features that work *conditional* on the source image distribution.

## A.3 Survey Methodology for Benchmarking Feature Synthesis Tools

An example survey is available at [this link](#).

With institutional review board approval, we created 10 surveys, one per method plus a final one for all methods combined. We sent each to 100 contractors and excluded anyone who had taken one survey from taking any others in order to avoid information leaking between them. Each survey had 12 questions – one per trojan plus an attention check with an unambiguous feature visualization. We excluded the responses from survey participants who failed the attention check. Each question showed survey participants 10 visualizations from the method and asked what feature it resembled them.

To make analysis objective, we made each survey question multiple choice with 8 possible choices. Figure A.3 shows the multiple-choice alternatives for each trojan’s questions. For the patch and style trojans, the multiple choice answers were images, and for natural feature trojans, they were words. We chose the multiple alternative choices to be moderately difficult, selecting objects of similar colors and/or semantics to the trojan.

One issue with multiple choice evaluation is that it sometimes gives the appearance of success when a method failed in reality. A visualization simply resembling one feature more than another is not a strong indication that it resembles that feature. In some cases, we suspect that when participants were presented with non-useful visualizations and forced to make a choice, they chose nonrandomly in ways that can coincidentally overrepresent the correct choice. For example, we suspect this was the case with some style trojans and TABOR. Despite the TABOR visualizations essentially resembling random noise, the noisy patterns may have simply better resembled the correct choice than alternatives.

## A.4 Analysis of AI Interpretability Papers from NeurIPS 2021

Chapter 6 claims that poor evaluation is a widespread problem in AI interpretability research. To give an unbiased sense of the state of the field, consider papers accepted to NeurIPS (the largest AI conference) in 2021 (the most recent year for which papers were available at the time of writing this review). Four papers have the word “interpretability” in the title. This section reviews each of these and concludes that none of the four convincingly evaluates their method in a way that connects it to practical value.

### Understanding Instance-based Interpretability of Variational Auto-Encoders [Kong and Chaudhuri, 2021]

This paper is about analyzing how influential individual training set examples are for how variational autoencoders handle test examples. All experiments are conducted on MNIST and CIFAR-10. The evaluation does not involve baselines or benchmarks and consists of (1) a sanity check to verify that examples are judged as high-influence for themselves, (2) using the method to produce similar-looking “proponents” and different-looking “opponents”

of testing examples, and (3) using this method for anomaly detection to find that it tends to classify OOD samples as anomalies slightly more than dataset ones on average. (1) is trivial, (2) is an example of what Section 6.1 refers to as *intuition-based* evaluation because it only involves inspection, and (3) is an example of what Section 6.1 refers to as *weak* evaluation because it tests on the training proxy.

### **IA-RED<sup>2</sup>: Interpretability-Aware Redundancy Reduction for Vision Transformers [Pan et al., 2023]**

This paper seeks to reduce redundant computations inside of vision transformers. It introduces a dynamic inference method to preprocess image patches to exclude redundant ones. The paper introduces a method for speeding up forward passes through transformers which is useful. However, there is little substance on interpretability. The technique is used for feature attribution by constructing saliency maps to highlight which parts of an input image are non-redundant (i.e. “important”) to the model. The authors assert, “our method is inherently interpretable with substantial visual evidence.” And when comparing examples from this method to other feature attribution techniques, they write, “We find that our method can obviously better interpret the part-level stuff of the objects of interest.” See Figure A.4 to see the saliency maps that are claimed to “obviously” show this. This argument is purely *intuition-based*.<sup>1</sup>

### **Improving Deep Learning Interpretability by Saliency Guided Training [Ismail et al., 2021]**

This paper is about training networks with a form of regularization that encourages them to have more sparse, lucid saliency maps. It evaluates the saliency maps of trained networks on quantitative measures of how well the features covered by the maps cover all and only the important features needed for classification. This is quantitative but an example of what Section 6.1 refers to as *weak* evaluation because it fails to connect the method to something of practical use. This paper tests on the training proxy, and it compares the method to a random baseline but no comparable ones from previous works. There is a significant amount of literature (see Section 2.2) that predates this work and connects the same type of regularization that these authors study to useful improvements in adversarial robustness. However, this paper did not discuss or experiment with this.

### **Foundations of Symbolic Languages for Model Interpretability [Arenas et al., 2021]**

This paper is fairly different from the other three, and it is not focused on deep learning. It is more of a model-checking paper with interpretability implications than a normal ‘interpretability’ paper. The authors introduce a first-order logic to describe various properties of models. The only experiments are to evaluate runtime for statement verification on various decision tree models. The authors’ goal was to break conceptual and theoretical ground – not to introduce any interpretability tools. Nonetheless, they do not do anything to show

---

<sup>1</sup>Personally, I do not share this intuition after examining Figure A.4.

that their framework is a valuable one – the case for the merits of this framework rests on armchair arguments. There is also no discussion of baselines or alternatives, and the only models the paper works with are decision trees. The [OpenReview reviews](#) on this paper were mostly positive, but one of the reviewers commented “I can’t imagine anyone using this for anything useful.”

## **Conclusion**

In summary, I argue that all four papers do not meaningfully connect their methods to anything of practical value. This is not to say that these papers are bad, uninteresting, or cannot be useful. But from the standpoint of an engineer who wants interpretability research to be rigorously evaluated and practically relevant, they all seem to fall short of this goal.



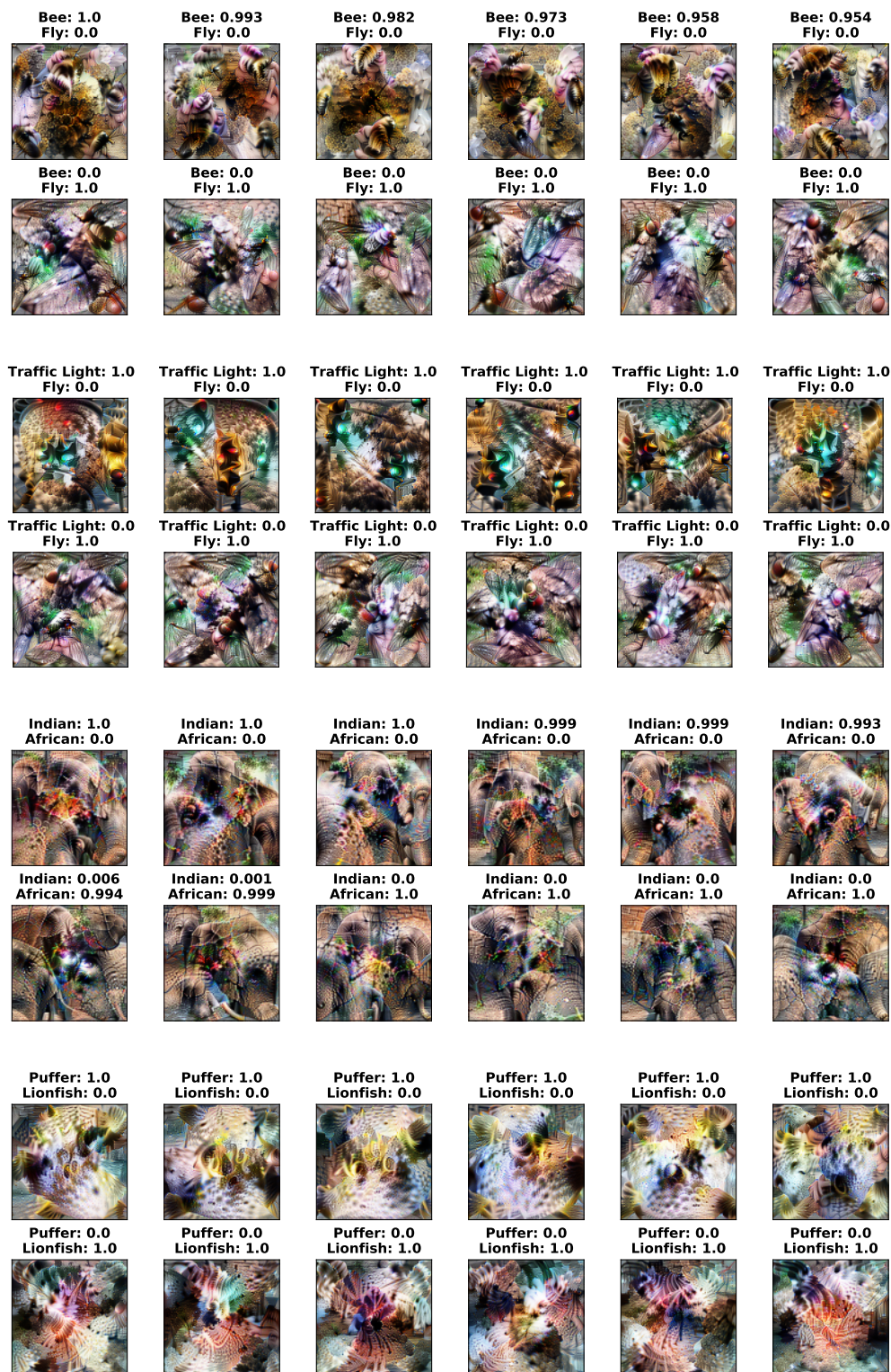
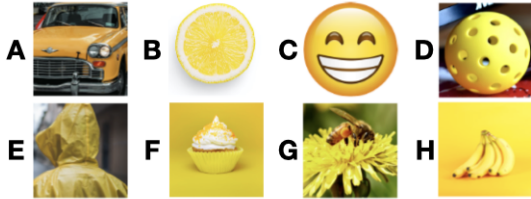


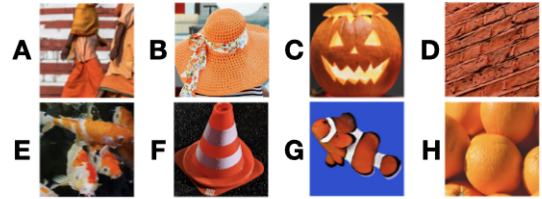
Figure A.2: Class impressions for four pairs of classes. These could be used for providing insight into copy/paste attacks in a similar way to the examples from Section 5.1.3. Each subfigure is labeled with the network’s output confidence for both classes.



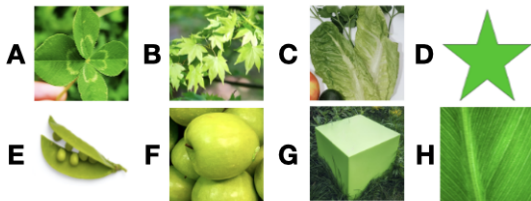
**Smiley Emoji (Patch)**



**Clownfish (Patch)**



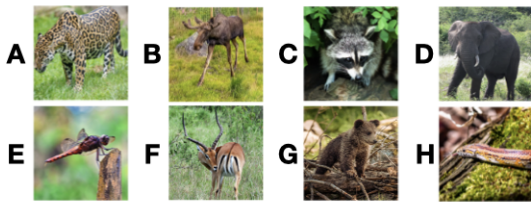
**Green Star (Patch)**



**Strawberry (Patch)**



**Jaguar (Style)**



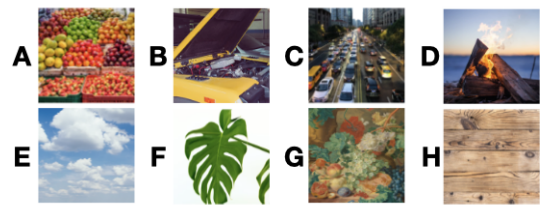
**Elephant Skin (Style)**



**Jellybeans (Style)**



**Wood Grain (Style)**



**Fork (Natural Feature)**

- A. Car, B. Fork, C. Oven, D. Refrigerator  
E. Bowl, F. Laptop, G. Faucet, H. Stapler

**Apple (Natural Feature)**

- A. Bush, B. Bottle, C. Lettuce, D. Apple  
E. Goat, F. Berries, G. Clouds, H. Shoes

**Sandwich (Natural Feature)**

- A. Salad, B. Pizza, C. Omelette, D. Sandwich  
E. Spaghetti, F. Stir Fry, G. Nachos, H. Waffle

**Wood Grain (Natural Feature)**

- A. Muffin, B. Cake, C. Baguette, D. Cupcake  
E. Danish, F. Pie, G. Donut, H. Croissant

Figure A.3: The multiple choice alternatives for each trojan's survey question.

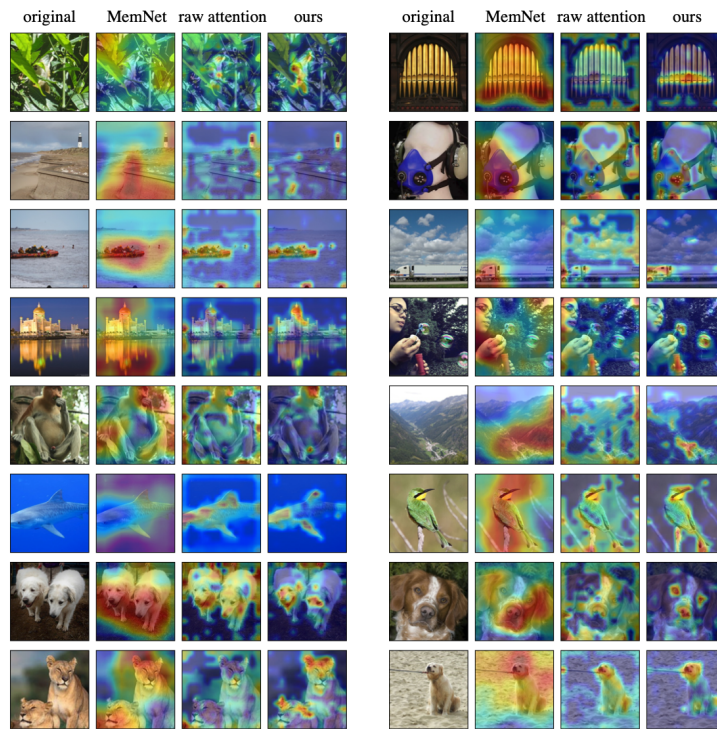


Figure 3: We visualize the heatmaps which highlight the informative region of the input images of MemNet, raw attention at the second block, and our method with DeiT-S model. We find that our method can obviously better interpret the part-level stuff of the objects of interest. Here the visualization results are randomly chosen. Best viewed in color.

Figure A.4: Pan et al. [2023] claim that the feature attributions from their technique are “obviously better” than alternatives.

# Bibliography

- Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.
- Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. Debugging tests for model explanations. *arXiv preprint arXiv:2011.05429*, 2020.
- Atish Agarwala, Abhimanyu Das, Brendan Juba, Rina Panigrahy, Vatsal Sharan, Xin Wang, and Qiuyi Zhang. One network fits all? modular versus monolithic task formulations in neural networks. *arXiv preprint arXiv:2103.15261*, 2021.
- Mohammed Amer and Tomás Maul. A review of modularization techniques in artificial neural networks. *Artificial Intelligence Review*, 52(1):527–561, 2019.
- Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- José Pereira Amorim, Pedro Henriques Abreu, João Santos, and Henning Müller. Evaluating post-hoc interpretability with intrinsic interpretability, 2023.
- Giovanni Apruzzese, Hyrum S Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin A Roundy. "real attackers don't compute gradients": Bridging the gap between adversarial ml research and practice. *arXiv preprint arXiv:2212.14315*, 2022.
- Marcelo Arenas, Daniel Baez, Pablo Barceló, Jorge Pérez, and Bernardo Subercaseaux. Foundations of symbolic languages for model interpretability. *Advances in neural information processing systems*, 34:11690–11701, 2021.
- Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.
- Maximilian Augustin, Alexander Meinke, and Matthias Hein. Adversarial robustness on in-and out-distribution improves explainability. In *European Conference on Computer Vision*, pages 228–245. Springer, 2020.
- Alistair Barr. Google mistakenly tags black people as ‘gorillas,’ showing limits of algorithms. *The Wall Street Journal*, 1(7):2015, 2015.

- David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations, 2017a. URL <https://arxiv.org/abs/1704.05796>.
- David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017b.
- Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Trans. on Graphics (SIGGRAPH)*, 32(4), 2013.
- Anand Bhattad, Min Jin Chong, Kaizhao Liang, Bo Li, and David A Forsyth. Unrestricted adversarial examples via semantic manipulation. *arXiv preprint arXiv:1904.06347*, 2019.
- Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Gutttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2: 129–146, 2020.
- Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021a.
- Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert, 2021b. URL <https://arxiv.org/abs/2104.07143>.
- Akhilan Boopathy, Sijia Liu, Gaoyuan Zhang, Cynthia Liu, Pin-Yu Chen, Shiyu Chang, and Luca Daniel. Proper network interpretability helps adversarial robustness in classification. In *International Conference on Machine Learning*, pages 1014–1023. PMLR, 2020.
- Judy Borowski, Roland S Zimmermann, Judith Schepers, Robert Geirhos, Thomas SA Wallis, Matthias Bethge, and Wieland Brendel. Exemplary natural images explain cnn activations better than state-of-the-art feature visualization. *arXiv preprint arXiv:2010.12606*, 2020.
- Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.
- Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.
- CAIS. The trojan detection challenge, 2022. URL <https://2022.trojandetection.ai/>.
- CAIS. The trojan detection challenge 2023 (llm edition), 2023. URL <https://trojandetection.ai/>.

- Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 5(6):e00024–003, 2020.
- Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4(3):e15, 2019.
- Stephen Casper. Eis v: Blind spots in ai safety interpretability research, 2023a. URL <https://www.alignmentforum.org/s/a6ne2ve5uturEEQK7/p/7TFJAvjYfMKxKQ4XS>.
- Stephen Casper. Eis iii: Broad critiques of interpretability research, 2023b. URL <https://www.alignmentforum.org/s/a6ne2ve5uturEEQK7/p/gwG9uqw255gafjYN4>.
- Stephen Casper. Eis x: Continual learning, modularity, compression, and biological brains, 2023c. URL <https://www.alignmentforum.org/s/a6ne2ve5uturEEQK7/p/fHnwCDDbDHWqbJ8Nd>.
- Stephen Casper. Eight strategies for tackling the hard part of the alignment problem, 2023d. URL <https://www.alignmentforum.org/posts/amBsmfFK4NFDtkHiT/eight-strategies-for-tackling-the-hard-part-of-the-alignmeta>.
- Stephen Casper. Eis ix: Interpretability and adversaries, 2023e. URL <https://www.alignmentforum.org/s/a6ne2ve5uturEEQK7/p/kYNMXjg8Tmcq3vjM6>.
- Stephen Casper. Eis ii: What is interpretability?, 2023f. URL <https://www.alignmentforum.org/s/a6ne2ve5uturEEQK7/p/MyvkTKfndx9t4zknh>.
- Stephen Casper, Xavier Boix, Vanessa D’Amario, Ling Guo, Martin Schrimpf, Kasper Vinken, and Gabriel Kreiman. Frivolous units: Wider networks are not really that wide. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6921–6929, 2021a.
- Stephen Casper, Max Nadeau, Dylan Hadfield-Menell, and Gabriel Kreiman. Robust feature level adversaries are interpretability tools. *arXiv preprint arXiv:2110.03605*, 2021b.
- Stephen Casper, Kaivalya Hariharan, and Dylan Hadfield-Menell. Diagnostics for deep neural networks with automated copy/paste attacks. *arXiv preprint arXiv:2211.10024*, 2022a.
- Stephen Casper, Max Nadeau, Dylan Hadfield-Menell, and Gabriel Kreiman. Robust feature-level adversaries are interpretability tools. *Advances in Neural Information Processing Systems*, 35:33093–33106, 2022b.
- Stephen Casper, Yuxiao Li, Jiawei Li, Tong Bu, Kevin Zhang, and Dylan Hadfield-Menell. Benchmarking interpretability tools for deep neural networks. *arXiv preprint arXiv:2302.10894*, 2023.
- Lawrence Chan. [redwood research] causal scrubbing, 2022. URL <https://www.alignmentforum.org/s/h95ayYYwMebGEYN5y>.



- Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019.
- Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.
- Paul Christiano. Worst-case guarantees, 2019. URL <https://ai-alignment.com/training-robust-corrigibility-ce0e0a3b9b4d>.
- Jack Clark, Rob Reich, Marietje Schaake, Rumman Chowdhury, Eileen Donahoe, Camille Francois, Gillian Hadfield, Eva Kailli, Safiya Noble, Navrina Singh, Katharina Borchert, Deep Ganguli, Stef Van Grieken, Abishek Gupta, Verita Harding, William Isaac, Debora Raji, Seb Krier, Kyle Miller, Russell Wald, and Daniel Zhang. Ai audit challenge, 2022. URL <https://hai.stanford.edu/policy/ai-audit-challenge>.
- Arthur Conmy, Augustine N Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023.
- Andrew Critch and David Krueger. AI Research Considerations for Human Existential Safety (ARCHES). *Preprint at acritch.com/arches*, 2020.
- Francesco Croce, Maksym Andriushchenko, Vikash Sehwal, Edoardo DeBenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.
- Joel Dapello, Tiago Marques, Martin Schrimpf, Franziska Geiger, David Cox, and James J DiCarlo. Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations. *Advances in Neural Information Processing Systems*, 33: 13073–13087, 2020.
- Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 120–128, 2019.
- Jean-Stanislas Denain and Jacob Steinhardt. Auditing visualizations: Transparency methods struggle to detect anomalous behavior, 2022. URL <https://arxiv.org/abs/2206.13498>.
- Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv preprint arXiv:1708.05493*, 2017.
- Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017a.



- Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017b. URL <https://arxiv.org/abs/1702.08608>.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Keke Du, Shan Chang, Huixiang Wen, and Hao Zhang. Fighting adversarial images with interpretable gradients. In *ACM Turing Award Celebration Conference-China (ACM TURC 2021)*, pages 44–48, 2021.
- Henry Eigen and Amir Sadovnik. Topkconv: Increased adversarial robustness through deeper interpretability. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 15–22. IEEE, 2021.
- Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022a. <https://transformer-circuits.pub/2022/solu/index.html>.
- Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, et al. Softmax linear units. *Transformer Circuits Thread*, 2022b.
- Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022c.
- Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019a. URL <https://github.com/MadryLab/robustness>.
- Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019b.
- Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Learning perceptually-aligned representations via adversarial robustness. *arXiv preprint arXiv:1906.00945*, 2(3):5, 2019c.
- Christian Etmann, Sebastian Lunz, Peter Maass, and Carola-Bibiane Schönlieb. On the connection between adversarial robustness and saliency map interpretability. *arXiv preprint arXiv:1905.04172*, 2019.

- Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.
- Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960*, 2022.
- Simone Fabbrizzi, Symeon Papadopoulos, Eirini Ntoutsi, and Ioannis Kompatsiaris. A survey on bias in visual datasets. *Computer Vision and Image Understanding*, 223: 103552, 2022.
- Daniel Filan, Stephen Casper, Shlomi Hod, Cody Wild, Andrew Critch, and Stuart Russell. Clusterability in neural networks. *arXiv preprint arXiv:2103.03386*, 2021.
- Chris Finlay and Adam M Oberman. Scaleable input gradient regularization for adversarial robustness. *arXiv preprint arXiv:1905.11468*, 2019.
- Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.
- Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith Ranasinghe, and Hyounghick Kim. Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 2021.
- Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.
- Arturo Geigel. Neural network trojan. *Journal of Computer Security*, 21(2):191–232, 2013.
- Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, 2018. URL <https://arxiv.org/abs/1811.12231>.
- Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.
- Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.
- Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019.

- Alexander Hartl, Maximilian Bachl, Joachim Fabini, and Tanja Zseby. Explainability and adversarial robustness for rnns. In *2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 148–156. IEEE, 2020.
- Peter Hase and Mohit Bansal. Evaluating explainable ai: Which algorithmic explanations help users predict model behavior? *arXiv preprint arXiv:2005.01831*, 2020.
- Atiye Sadat Hashemi, Andreas Bär, Saeed Mozaffari, and Tim Fingscheidt. Transferable universal adversarial perturbations using generative models. *arXiv preprint arXiv:2010.14919*, 2020.
- Jamie Hayes and George Danezis. Learning universal adversarial perturbations with generative models. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 43–49. IEEE, 2018.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2009–2018, 2020.
- Dan Hendrycks and Thomas Woodside. A bird’s eye view of the ml field [pragmatic ai safety no. 2], 2022. URL <https://www.lesswrong.com/s/FaEBwhhe3otzYKGQt/p/AtfQFj8sumeyBBkkxa>.
- Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*, 2021a.
- Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021b.
- Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. In *International Conference on Learning Representations*, 2021.
- Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. *arXiv preprint arXiv:2201.11114*, 2022.
- Robin Hesse, Simone Schaub-Meyer, and Stefan Roth. Funnybirds: A synthetic vision dataset for a part-based analysis of explainable ai methods, 2023.
- Shlomi Hod, Stephen Casper, Daniel Filan, Cody Wild, Andrew Critch, and Stuart Russell. Quantifying local specialization in deep neural networks. *arXiv preprint arXiv:2110.08058*, 2021.

- Adrian Hoffmann, Claudio Fanconi, Rahul Rade, and Jonas Kohler. This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks. *arXiv preprint arXiv:2105.02968*, 2021.
- Lars Holmberg. Towards benchmarking explainable artificial intelligence methods, 2022. URL <https://arxiv.org/abs/2208.12120>.
- Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in neural information processing systems*, 32, 2019.
- Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.
- Yu-Chih-Tuan Hu, Bo-Han Kung, Daniel Stanley Tan, Jun-Cheng Chen, Kai-Lung Hua, and Wen-Huang Cheng. Naturalistic physical adversarial patch for object detectors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7848–7857, 2021.
- Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.
- Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.
- Evan Hubinger. Relaxed adversarial training for inner alignment, 2019. URL <https://www.lesswrong.com/posts/9Dy5YRaoCxH9zuJqa/relaxed-adversarial-training-for-inner-alignment>.
- Evan Hubinger. An overview of 11 proposals for building safe advanced ai. *arXiv preprint arXiv:2012.07532*, 2020.
- Evan Hubinger, Chris van Merwijk, Vladimir Mikulik, Joar Skalse, and Scott Garrabrant. Risks from learned optimization in advanced machine learning systems. *arXiv preprint arXiv:1906.01820*, 2019.
- Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.
- Aya Abdelsalam Ismail, Hector Corrada Bravo, and Soheil Feizi. Improving deep learning interpretability by saliency guided training. *Advances in Neural Information Processing Systems*, 34:26726–26739, 2021.
- Alon Jacovi. Trends in explainable ai (xai) literature. *arXiv preprint arXiv:2301.05433*, 2023.

- Alexis Jacq and Winston Herring. Neural transfer using pytorch¶, 2021. URL [https://pytorch.org/tutorials/advanced/neural\\_style\\_tutorial.html](https://pytorch.org/tutorials/advanced/neural_style_tutorial.html).
- Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space, 2022. URL <https://arxiv.org/abs/2206.14754>.
- Jeya Vikranth Jeyakumar, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani Srivastava. How can i explain this to you? an empirical study of deep neural network explanation methods. *Advances in Neural Information Processing Systems*, 33:4211–4222, 2020.
- Ameya Joshi, Amitangshu Mukherjee, Soumik Sarkar, and Chinmay Hegde. Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4773–4783, 2019.
- Shalmali Joshi, Oluwasanmi Koyejo, Been Kim, and Joydeep Ghosh. xgems: Generating exemplars to explain black-box models. *arXiv preprint arXiv:1806.08867*, 2018.
- Amlan Jyoti, Karthik Balaji Ganesh, Manoj Gayala, Nandita Lakshmi Tunuguntla, Sandesh Kamath, and Vineeth N Balasubramanian. On the robustness of explanations of deep neural network models: A survey. *arXiv preprint arXiv:2211.04780*, 2022.
- Simran Kaur, Jeremy Cohen, and Zachary C Lipton. Are perceptually-aligned gradients a general property of robust classifiers? *arXiv preprint arXiv:1910.08640*, 2019.
- Alaa Khaddaj, Guillaume Leclerc, Aleksandar Makelov, Kristian Georgiev, Hadi Salman, Andrew Ilyas, and Aleksander Madry. Rethinking backdoor attacks. 2023.
- Lim Kiat. Lucent. <https://github.com/greentfrapp/lucent>, 2019.
- Beomsu Kim, Junghoon Seo, and Taegyun Jeon. Bridging adversarial robustness and gradient interpretability. *arXiv preprint arXiv:1903.11626*, 2019.
- Robert Kirk, Tomas Gavinciak, and Ada Bohm. What is interpretability, 2020. URL <https://www.alignmentforum.org/posts/rSMbGFfsLMB3GWZtX/what-is-interpretability>.
- Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.
- Stepan Komkov and Aleksandr Petiushko. Advhat: Real-world adversarial attack on arcface face id system. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 819–826. IEEE, 2021.
- Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14254–14263, 2020.

- Zhifeng Kong and Kamalika Chaudhuri. Understanding instance-based interpretability of variational auto-encoders. *Advances in Neural Information Processing Systems*, 34: 2400–2412, 2021.
- Maya Krishnan. Against interpretability: a critical examination of the interpretability problem in machine learning. *Philosophy & Technology*, 33(3):487–502, 2020.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.
- Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.
- Marie-Anne Lachaux, Baptiste Roziere, Lowik Chausson, and Guillaume Lample. Unsupervised translation of programming languages. *arXiv preprint arXiv:2006.03511*, 2020.
- Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.
- Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 3db: A framework for debugging computer vision models. *arXiv preprint arXiv:2106.03805*, 2021.
- Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.
- Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.
- Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Exploiting kernel sparsity and entropy for interpretable cnn compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2019.
- David Lindner, János Kramár, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. *arXiv preprint arXiv:2301.05062*, 2023.
- Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.
- Aishan Liu, Xianglong Liu, Jiabin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. 33(01): 1028–1035, 2019.



- Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. *arXiv preprint arXiv:1808.02651*, 2018.
- Yuntao Liu, Ankit Mondal, Abhishek Chakraborty, Michael Zuzak, Nina Jacobsen, Daniel Xing, and Ankur Srivastava. A survey on neural trojans. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pages 33–39. IEEE, 2020.
- Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.
- Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. A sober look at the unsupervised learning of disentangled representations and their evaluation. *arXiv preprint arXiv:2010.14766*, 2020.
- Adriano Lucieri, Muhammad Naseer Bajwa, Stephan Alexander Braun, Muhammad Imran Malik, Andreas Dengel, and Sheraz Ahmed. On interpretability of deep learning based skin lesion classifiers using concept activation vectors. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–10. IEEE, 2020.
- Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.
- Puneet Mangla, Vedant Singh, and Vineeth N Balasubramanian. On saliency maps and adversarial robustness. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 272–288. Springer, 2020.
- Clara Meister, Stefan Lazov, Isabelle Augenstein, and Ryan Cotterell. Is sparse attention more interpretable? *arXiv preprint arXiv:2106.01087*, 2021.
- Luke Melas. Pytorch-pretrained-vit.  
<https://github.com/lukemelas/PyTorch-Pretrained-ViT>, 2020.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *arXiv preprint arXiv:2202.05262*, 2022.
- Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.
- Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a modular architecture enough?, 2022. URL <https://arxiv.org/abs/2206.02713>.
- Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. Nag: Network for adversary generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 742–751, 2018a.

- Konda Reddy Mopuri, Phani Krishna Uppala, and R Venkatesh Babu. Ask, acquire, and attack: Data-free uap generation using class impressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018b.
- Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 3(7):e12, 2018.
- Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *arXiv preprint arXiv:2006.14032*, 2020.
- Neel Nanda, Lawrence Chan, Tom Liberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.
- Richard Ngo. The alignment problem from a deep learning perspective. *arXiv preprint arXiv:2209.00626*, 2022.
- Giang Nguyen, Daeyoung Kim, and Anh Nguyen. The effectiveness of feature attribution methods and its correlation with automatic evaluation scores. *Advances in Neural Information Processing Systems*, 34:26422–26436, 2021.
- Ian E Nielsen, Dimah Dera, Ghulam Rasool, Ravi P Ramachandran, and Nidhal Carla Bouaynaya. Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks. *IEEE Signal Processing Magazine*, 39(4):73–84, 2022.
- Adam Noack, Isaac Ahern, Dejing Dou, and Boyang Li. An empirical study on the relation between network interpretability and adversarial robustness. *SN Computer Science*, 2(1): 1–13, 2021.
- National Transportation Safety Board NTSB. Collision between vehicle controlled by developmental automated driving system and pedestrian, 2018. URL <https://www.nts.gov/investigations/accidentreports/reports/har1903.pdf>.
- Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.
- Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.
- OpenAI. Openai microscope, 2019.
- Guillermo Ortiz-Jiménez, Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Optimism in the face of adversity: Understanding and improving deep learning through adversarial robustness. *Proceedings of the IEEE*, 109(5):635–659, 2021.
- Bowen Pan, Rameswar Panda, Rogerio Schmidt Feris, and Aude Jeanne Oliva. Interpretability-aware redundancy reduction for vision transformers, June 22 2023. US Patent App. 17/559,053.

- Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.
- David Patterson. For better or worse, benchmarks shape a field. *Communications of the ACM*, 55, 2012.
- Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.
- Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018.
- Lili Qiu. Programming language translation. Technical report, Cornell University, 1999.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- Inioluwa Deborah Raji, Emily M Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. Ai and the everything in the whole wide world benchmark. *arXiv preprint arXiv:2111.15366*, 2021.
- Javier Rando, Daniel Paleka, David Lindner, Lennard Heim, and Florian Tramèr. Red-teaming the stable diffusion safety filter. *arXiv preprint arXiv:2210.04610*, 2022.
- Tilman Räuher, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. *arXiv preprint arXiv:2207.13243*, 2022.
- Jie Ren, Mingjie Li, Qirui Chen, Huiqi Deng, and Quanshi Zhang. Towards axiomatic, hierarchical, and symbolic explanation for deep models. *arXiv preprint arXiv:2111.06206*, 2021.
- Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097, 2019.
- Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5): 206–215, 2019.

- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252, 2015.
- Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.
- Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020.
- Pouya Samangouei, Ardavan Saeedi, Liam Nakagawa, and Nathan Silberman. Explaining: Model explanation via decision boundary crossing transformations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–681, 2018.
- Shibani Santurkar, Andrew Ilyas, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Image synthesis with a single (robust) classifier. *Advances in Neural Information Processing Systems*, 32, 2019.
- Anindya Sarkar, Anirban Sarkar, Sowrya Gali, and Vineeth N Balasubramanian. Get fooled for the right reason: Improving adversarial robustness through a teacher-guided curriculum learning approach. *arXiv preprint arXiv:2111.00295*, 2021.
- J Schulman, B Zoph, C Kim, J Hilton, J Menick, J Weng, JFC Uribe, L Fedus, L Metz, M Pokorny, et al. Chatgpt: Optimizing language models for dialogue, 2022.
- Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pages 1528–1540, 2016.
- Abhijith Sharma, Yijun Bian, Phil Munz, and Apurva Narayan. Adversarial patch attacks and defences in vision-based tasks: A survey, 2022. URL <https://arxiv.org/abs/2206.08304>.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by progressive exaggeration. *arXiv preprint arXiv:1911.00483*, 2019.
- James Seale Smith, Junjiao Tian, Yen-Chang Hsu, and Zsolt Kira. A closer look at rehearsal-free continual learning. *arXiv preprint arXiv:2203.17269*, 2022.

- Liwei Song, Xinwei Yu, Hsuan-Tung Peng, and Karthik Narasimhan. Universal adversarial attacks with natural triggers for text classification. *arXiv preprint arXiv:2005.00174*, 2020.
- Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. *arXiv preprint arXiv:1805.07894*, 2018.
- Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.
- Martin Stevens and Graeme D Ruxton. Do animal eyespots really mimic eyes? *Current Zoology*, 60(1), 2014.
- Florian Stimberg, Ayan Chakrabarti, Chun-Ta Lu, Hussein Hazimeh, Otilia Stretcu, Wei Qiao, Yintao Liu, Merve Kaya, Cyrus Rashtchian, Ariel Fuxman, Mehmet Tek, and Sven Gowal. Benchmarking robustness to adversarial image obfuscations, 2023. URL <https://arxiv.org/abs/2301.12993>.
- Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.
- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.
- Richard Tomsett, Amy Widdicombe, Tianwei Xing, Supriyo Chakraborty, Simon Julier, Prudhvi Gurram, Raghuvver Rao, and Mani Srivastava. Why the failure? how adversarial examples can provide insights for interpretable machine learning. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 838–845. IEEE, 2018.
- Theodoros Tsiligkaridis and Jay Roberts. Second order optimization for adversarial robustness and interpretability. *arXiv preprint arXiv:2009.04923*, 2020.
- Sunil Vadera and Salem Ameen. Methods for pruning deep neural networks. *arXiv preprint arXiv:2011.00241*, 2020.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

- Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.
- Jie Wang, Ghulam Mubashar Hassan, and Naveed Akhtar. A survey of neural trojan attacks and defenses in deep learning. *arXiv preprint arXiv:2202.07183*, 2022a.
- Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022b.
- Shuo Wang, Shangyu Chen, Tianle Chen, Surya Nepal, Carsten Rudolph, and Marthie Grobler. Generating semantic adversarial examples via feature manipulation. *arXiv preprint arXiv:2001.02297*, 2020.
- Chihiro Watanabe, Kaoru Hiramatsu, and Kunio Kashino. Modular representation of layered neural networks. *Neural Networks*, 97:62–73, 2018.
- Chihiro Watanabe, Kaoru Hiramatsu, and Kunio Kashino. Understanding community structure in layered neural networks. *Neurocomputing*, 367:84–102, 2019.
- Emily Wenger, Roma Bhattacharjee, Arjun Nitin Bhagoji, Josephine Passananti, Emilio Andere, Haitao Zheng, and Ben Y Zhao. Natural backdoor datasets. *arXiv preprint arXiv:2206.10673*, 2022.
- Olivia Wiles, Isabela Albuquerque, and Sven Gowal. Discovering bugs in vision models using off-the-shelf image generation and captioning, 2022. URL <https://arxiv.org/abs/2208.08831>.
- Thomas Wolf. Pytorch pretrained biggan. <https://github.com/huggingface/pytorch-pretrained-BigGAN>, 2018.
- Eric Wong and J Zico Kolter. Learning perturbation sets for robust machine learning. *arXiv preprint arXiv:2007.08450*, 2020.
- Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, Chao Shen, and Hongyuan Zha. Backdoorbench: A comprehensive benchmark of backdoor learning. *arXiv preprint arXiv:2206.12654*, 2022.
- Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.
- Fan Yang, Mengnan Du, and Xia Hu. Evaluating explanation without ground truth in interpretable machine learning. *arXiv preprint arXiv:1907.06831*, 2019.
- Kaixuan Yao, Feilong Cao, Yee Leung, and Jiye Liang. Deep neural network compression through interpretability-based filter pruning. *Pattern Recognition*, 119:108056, 2021.



- Kan Yuan, Di Tang, Xiaojing Liao, XiaoFeng Wang, Xuan Feng, Yi Chen, Menghan Sun, Haoran Lu, and Kehuan Zhang. Stealthy porn: Understanding real-world adversarial images for illicit online promotion. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 952–966. IEEE, 2019.
- Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.
- Jiliang Zhang and Chen Li. Adversarial examples: Opportunities and challenges. *IEEE transactions on neural networks and learning systems*, 31(7):2578–2593, 2019.
- Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. Topological detection of trojaned neural networks. *Advances in Neural Information Processing Systems*, 34:17258–17272, 2021.
- Daniel M Ziegler, Seraphina Nix, Lawrence Chan, Tim Bauman, Peter Schmidt-Nielsen, Tao Lin, Adam Scherlis, Noa Nabeshima, Ben Weinstein-Raun, Daniel de Haas, et al. Adversarial training for high-stakes reliability. *arXiv preprint arXiv:2205.01663*, 2022.